Image filtering & noise suppression

Jan Kybic

http://cmp.felk.cvut.cz/~kybic

kybic@fel.cvut.cz

September 2011

with contributions and slides from Václav Hlaváč, Tomáš Svoboda, Alan Peters

Introduction

Noise

Spatial filtering

Frequency domain filtering

Non-linear filtering

Filtering applications

Denoising by linear filtering

Wavelets

Filtering

Image processing operation

- Uses information from more than one pixel
- Spatially invariant (only local information)
- Does not change geometry (position of objects)

Motivation & Examples

- Noise suppression
- Blurring, sharpening
- Illumination inhomogeneity suppression,
  local contrast improvement
- Object detection

Noise

Sources:

- Photon noise
- Sensor noise
- Film grain
- Thermal noise in electronics
- Transmission noise
- Quantization noise

Noise reduction:

- Bigger sensors
- Cooled sensors and electronics
- Long exposures, repeated acquisitions
- Image processing

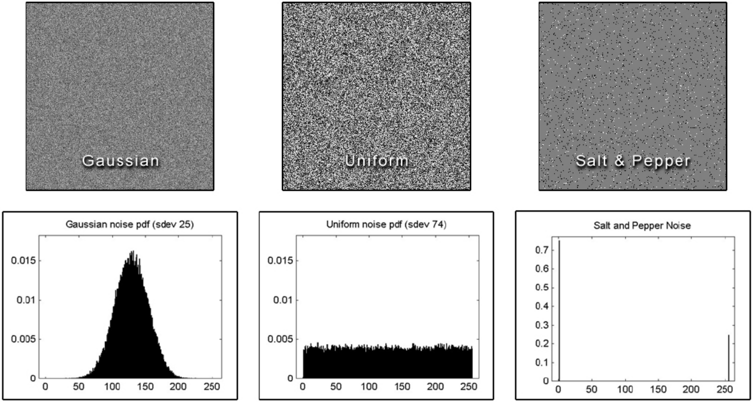
Noise properties

- Additive $f_n(\mathbf{x}) = f(\mathbf{x}) + u(\mathbf{x})$, multiplicative $f_n(\mathbf{x}) = f(\mathbf{x})u(\mathbf{x})$
- Gaussian, uniform, salt&pepper *(histogram, formulas $p_u(u)$)*

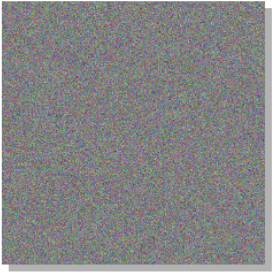
$$p_u(u) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(u-\mu_u)^2}{2\sigma^2}}$$

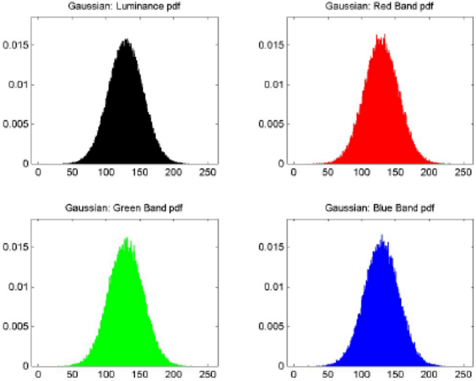
- Identically distributed $\times$ spatially variant
  $p_u(u \mid \mathbf{x}) \stackrel{?}{=} p_u(u)$
- Independent ($\Rightarrow$ uncorrelated) $\times$ correlated
  $\operatorname{cov}\big(u(\mathbf{x}), u(\mathbf{y})\big) = \mathrm{E}\big[(u(\mathbf{x}) - \mu_u(\mathbf{x}))(u(\mathbf{y}) - \mu_u(\mathbf{y}))\big] \stackrel{?}{=} c\delta(\mathbf{x} - \mathbf{y})$
- zero mean, $\mu_u = \bar{u} = \mathrm{E}[u] = 0$
- independent identically distributed (i.i.d.) normal noise

Uncorrelated noise examples

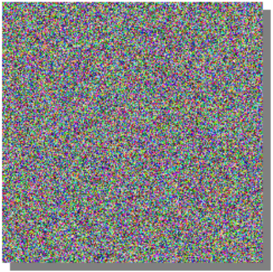


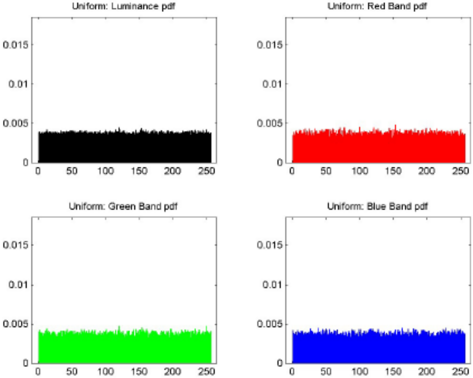
Uncorrelated noise examples





Uncorrelated noise examples





Multiple image averaging

Acquire N images of the same scene. Assume i.i.d. additive zero-mean Gaussian noise with variance σ^2.

$$f_i(\mathbf{x}) = f(\mathbf{x}) + u(\mathbf{x})$$

- each $f_i(\mathbf{x})$ has $\mathrm{E}\big[f_i(\mathbf{x})\big] = f(\mathbf{x})$ and $\mathrm{var}\big[f_i(\mathbf{x}_i)\big] = \sigma^2$
- Calculate average value $\bar{f}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$
- $\bar{f}$ is an unbiased estimator of f

$$\mathrm{E}\big[\bar{f}(\mathbf{x})\big] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}\big[f_i(\mathbf{x})\big] = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}) = f(\mathbf{x})$$

Multiple image averaging

Acquire N images of the same scene. Assume i.i.d. additive zero-mean Gaussian noise with variance σ^2.

$$f_i(\mathbf{x}) = f(\mathbf{x}) + u(\mathbf{x})$$

- each $f_i(\mathbf{x})$ has $\mathrm{E}\big[f_i(\mathbf{x})\big] = f(\mathbf{x})$ and $\mathrm{var}\big[f_i(\mathbf{x}_i)\big] = \sigma^2$
- Calculate average value $\bar{f}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$
- Variance is decreased N times, standard deviation $\sqrt{N}$ times

$$\mathrm{var}\big[\bar{f}(\mathbf{x})\big] = \frac{1}{N^2}\,\mathrm{var}\left[\sum_{i=1}^{N} f_i(\mathbf{x})\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{var}\big[f_i(\mathbf{x})\big]$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\sigma^2 = \frac{\sigma^2}{N}$$

- $\rightarrow \mathrm{stdev}\big[\bar{f}(\mathbf{x})\big] = \mathrm{stdev}\big[u\big]/\sqrt{N} = \sigma/\sqrt{N}$

Multiple image averaging
Example

noisy

Multiple image averaging

Example

average

Multiple image averaging

Example

Multiple image averaging

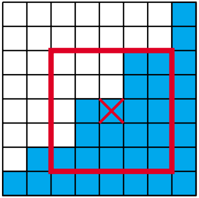
Example

Beyond multiple image averaging

- Mostly only a single image available
- We can use spatial redundancy (neighborhood pixels are similar)
- Local processing
- Distinguish between noise and image features (e.g. edges)

Local filtering

- Replace a value of the image function (pixel) by a new one computed from the immediate neighbourhood.
- Linear $\times$ nonlinear
- Shift-invariant or not



- spatial relationships are important
- small neighborhood $\rightarrow$ fast
- linear $\rightarrow$ fast
- averaging suppresses Gaussian noise but causes blurring
- robust statistics can be applied

Local filtering (2)

Idea: Output is a function of a pixel value and those of its neighbours.

Example for a 3×3 region.

$$g(x,y) = \mathrm{Op}\begin{pmatrix} f(x-1,y-1) & f(x,y-1) & f(x+1,y-1) \\ f(x-1,y) & f(x,y) & f(x+1,y) \\ f(x-1,y+1) & f(x,y+1) & f(x+1,y+1) \end{pmatrix}$$

Possible operations: sum, average, weighted sum, min, max, median . . .

Linear shift invariant filtering

- linearity $\alpha\mathcal{L}(f_1) + \beta\mathcal{L}(f_2) = \mathcal{L}\big(\alpha f_1 + \beta f_2\big)$
- shift-invariance $\big(\mathcal{L}(f)\big)(\mathbf{x} + \mathbf{t}) = \mathcal{L}\big(f(\mathbf{x} + \mathbf{t})\big)$
- $\Leftrightarrow$ can be expressed as a **convolution** with kernel h

$$g = \mathcal{L}(f) = f * h$$

$$g(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-k, y-l)h(k, l)\mathrm{d}k\mathrm{d}l$$

$$g(x, y) = \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} f(x-k, y-l)h(k, l)$$

- simplest, fastest, models well the acquisition process and its inverse, optimum denoising in the Gaussian case

Spatial filtering by masks

- Very common neighbour operation is per-element multiplication with a set of weights and sum together.
- Set of weights is often called **mask** or **kernel**.

Local neighbourhood

f(x-1,y-1)	f(x,y-1)	f(x+1,y-1)
f(x-1,y)	f(x,y)	f(x+1,y)
f(x-1,y+1)	f(x,y+1)	f(x+1,y+1)

mask

w(-1,-1)	w(0,-1)	w(+1,-1)
w(-1,0)	w(0,0)	w(+1,0)
w(-1,+1)	w(0,+1)	w(+1,+1)

$$g(x,y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} w(k,l) f(x+k, y+l)$$

2D convolution

- Spatial filtering is often referred to as **convolution**.
- We say, we **convolve** the image by a kernel or mask.
- Though, it is not the same. Convolution uses a flipped kernel.

Local neighbourhood

f(x-1,y-1)	f(x,y-1)	f(x+1,y-1)
f(x-1,y)	f(x,y)	f(x+1,y)
f(x-1,y+1)	f(x,y+1)	f(x+1,y+1)

mask

w(+1,+1)	w(0,+1)	w(-1,+1)
w(+1,0)	w(0,0)	w(-1,0)
w(+1,-1)	w(0,-1)	w(-1,-1)

$$g(x, y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} w(k, l) f(x - k, y - l)$$

Smoothing

Output value is computed as an average of the input value and its neighbourhood.

- Advantage: less noise
- Disadvantage: blurring
- Any kernel with all positive weights causes **smoothing** or **blurring**
- They are called **low-pass filters**

Averaging (preserves constants):

$$g(x, y) = \frac{\sum_k \sum_l w(k, l) f(x + k, y + l)}{\sum_k \sum_l w(k, l)}$$

Smoothing kernels

Can be of any size, any shape

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix},$$

$$h = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Averaging ones($n \times n$) — increasing mask size

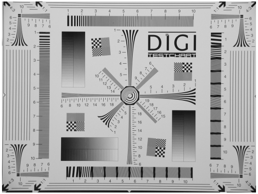
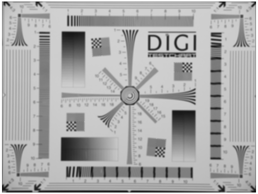
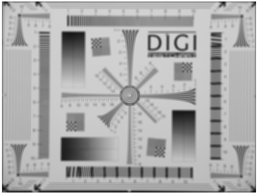


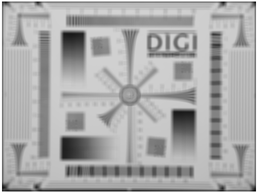
image 1024×768



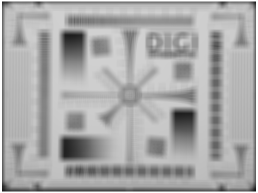
7×7



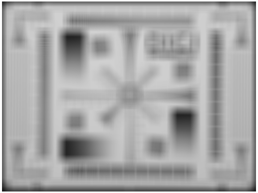
11×11



15×15

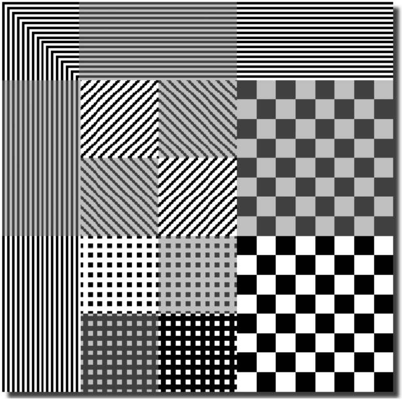


29×29



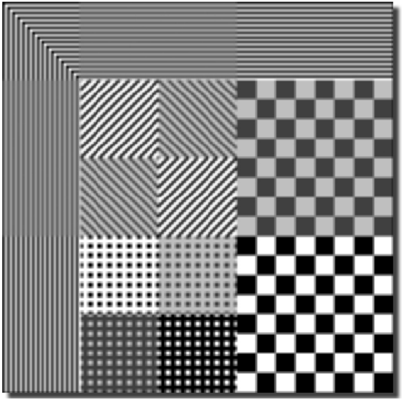
43×43

More smoothing examples

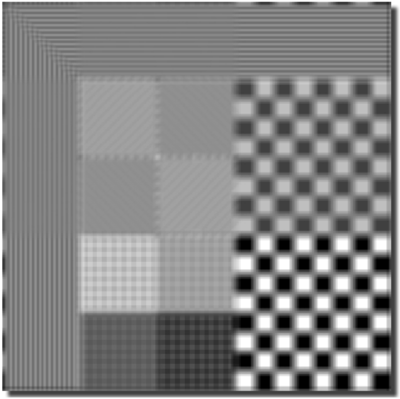
original

More smoothing examples




3×3 kernel

More smoothing examples



9×9 kernel

Gaussian filter

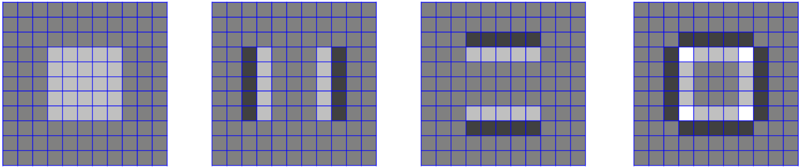
$$h(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} \left|\mathbf{C}\right|^{-\frac{1}{2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}-\boldsymbol{\mu})^T}$$

- Separable
- Rotation invariant (for $\mathbf{C} = c\mathbf{I}$)
- Smooth in both space and frequency
- Well approximates many natural processes
- Central limit theorem justification

Implementation

- Infinite support
- Truncate = FIR approximation
- IIR (moving average) approximation (Deriche)
- Binomial filter = repeated convolution with $[1 \; \ldots \; 1]$
- Fourier domain implementation

Derivative (difference) filters

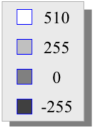


$I(r,c)$

$2I(r,c) - I(r,c-1) - I(r,c+1)$

$2I(r,c) - I(r-1,c) - I(r+1,c)$

$4I(r,c) - I(r-1,c) - I(r+1,c) - I(r,c-1) - I(r,c+1)$

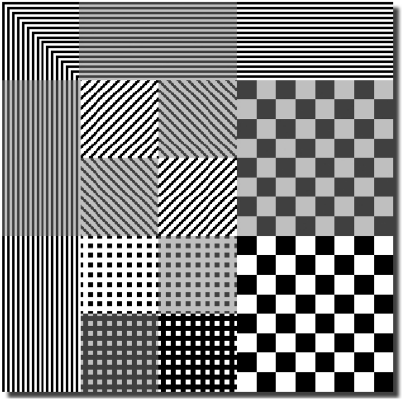


-1	2	-1

	-1	
	2	
	-1	

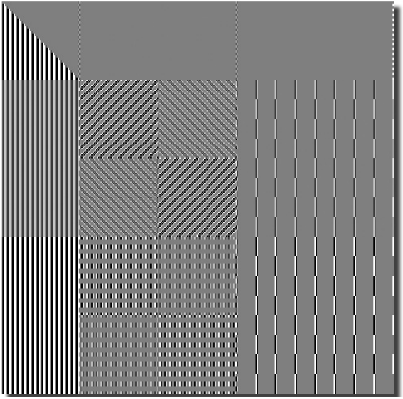
	-1	
-1	4	-1
	-1	

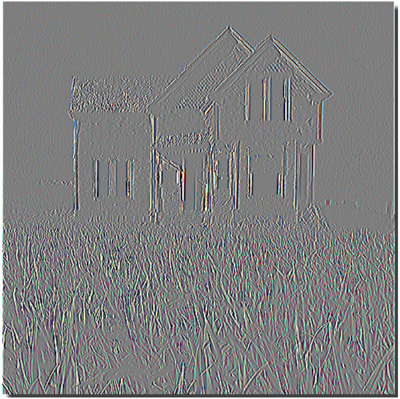
Derivative examples



original

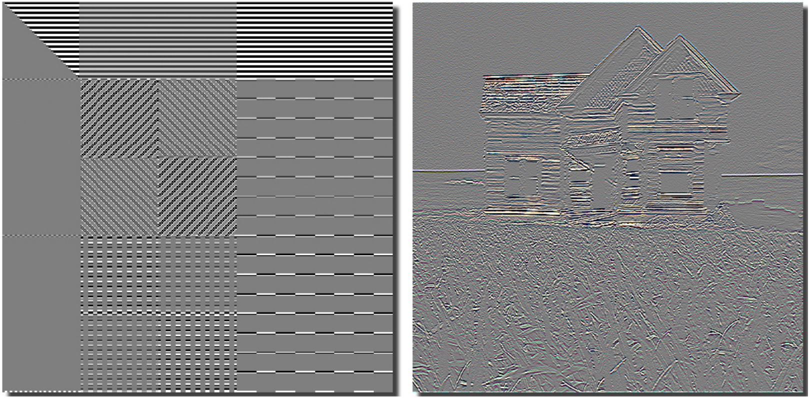
Derivative examples





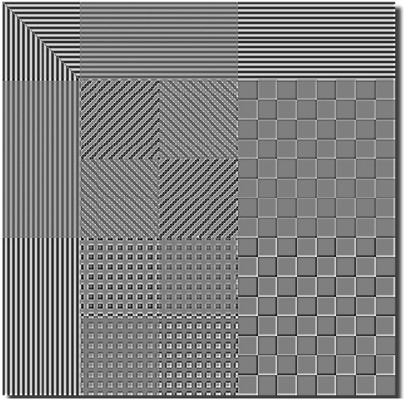
horizontal differences

Derivative examples



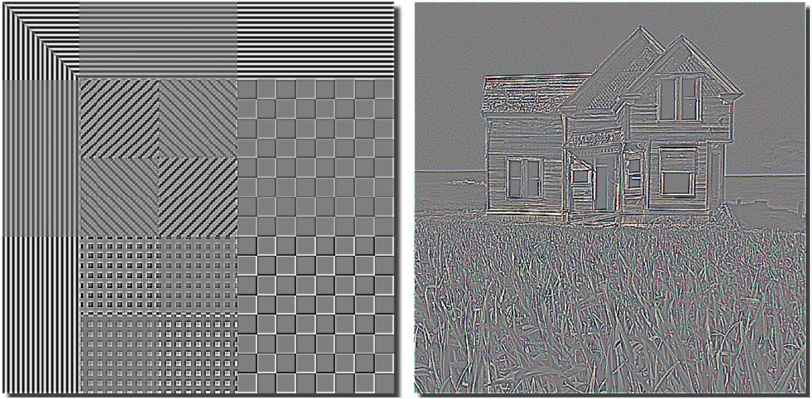
vertical differences

Derivative examples



horizontal & vertical differences

Derivative examples



horizontal & vertical & diagonal differences

Frequency domain filtering

- LTI spatial domain filtering — convolution
- Frequency domain filtering
  - Forward transform (FFT)
  - Filtering — multiplication
  - Inverse transform (iFFT)
- **Motivation** — efficiency, interpretation
- Other transforms possible (wavelet, Hadamard. . .)

2D Fourier transform

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)}dxdy$$

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(xu+yv)}dudv$$

- A *sufficient* condition for existence is absolute integrability.
- Basis functions are sin and cos thanks to $e^{jz} = \cos z + j \sin z$
- Various notations: $\hat{f}$, $\mathcal{F}_{u,v}(f) = F(u,v)$

Discrete Fourier transform

$$F(\xi,\eta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-2\pi j(x\xi+y\eta)}\, dx\, dy$$

Problem: Images are discretized and restricted in space...

- Periodicity $\rightarrow$ Fourier series, discretized frequencies $(u = 0, 1, \dots)$

$$F_s(u,v) = \frac{1}{N_x N_y} \int_0^{N_x}\int_0^{N_y} f(x,y)\, e^{-2\pi j(\frac{xu}{N_x}+\frac{yv}{N_y})}\, dx\, dy$$

- Integration using P0 interpolation, ideal sampling $(h = 1) \rightarrow$ DFT:

$$F_d(u,v) = \frac{1}{N_x N_y} \sum_{x=0}^{N_x-1}\sum_{y=0}^{N_y-1} f(x,y)\, e^{-2\pi j(\frac{xu}{N_x}+\frac{yv}{N_y})}$$

- Inverse transform (IDFT):

$$f(x,y) = \sum_{u=0}^{N_x-1}\sum_{v=0}^{N_y-1} F_d(x,y)\, e^{2\pi j(\frac{xu}{N_x}+\frac{yv}{N_y})}$$

Discrete Fourier transform

- Integration using P0 interpolation, ideal sampling ($h = 1$) $\rightarrow$ DFT:

$$F_d(u,v) = \frac{1}{N_x N_y} \sum_{x=0}^{N_x-1} \sum_{y=0}^{N_y-1} f(x,y)\; e^{-2\pi j\left(\frac{xu}{N_x}+\frac{yv}{N_y}\right)}$$

- Inverse transform (IDFT):

$$f(x,y) = \sum_{u=0}^{N_x-1} \sum_{v=0}^{N_y-1} F_d(x,y)\; e^{2\pi j\left(\frac{xu}{N_x}+\frac{yv}{N_y}\right)}$$

- Separability, $\mathcal{F}_x \mathcal{F}_y = \mathcal{F}_{xy}$
- Fast Fourier Tranform (FFT) with complexity $O\big(N_x N_y \log N_y N_x\big)$. Choose $N = 2^n$. Matlab `fft`,`fft2`, `ifft`, `ifft2`.

  $\rightarrow$ **Fast implementation**
- Different normalizations ($N_x N_y$).

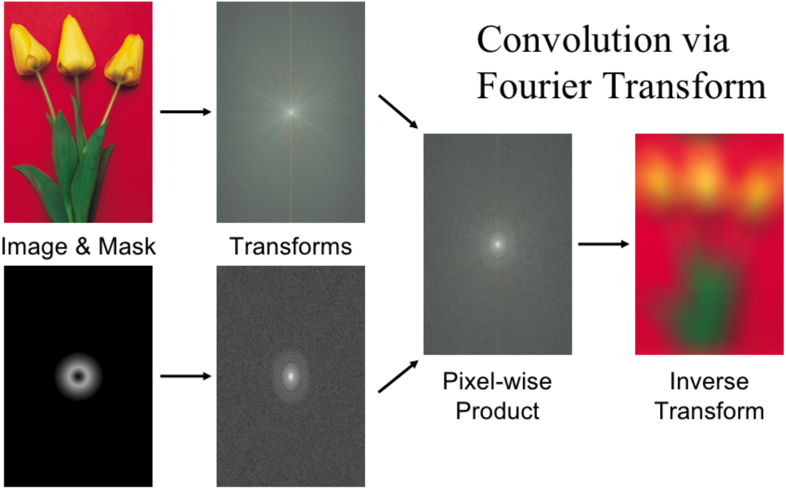
Convolution theorem

- Functions $f(x, y)$ and $g(x, y)$ with FT $F(u, v)$ and $G(u, v)$.
- $\mathcal{F}(f * g) = F \cdot G$
- $\mathcal{F}(f \cdot g) = F * G$

Notes

- Transforms must be normalized.
- Periodic boundary conditions are implied if DFT is used.
- Computational savings for large kernels,
  $O(N_\mathrm{F} \log N_\mathrm{F})$ instead of $N_\mathrm{F} N_\mathrm{G}$.
- We usually display $\log |F|$ but the filter must be applied to F.

Convolution via Fourier Transform



FT convolution in Matlab

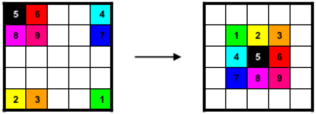
To calculate $J = I * h$, i.e. $\mathcal{F}(J) = \mathcal{F}(I)\,\mathcal{F}(h)$ or $\hat{J} = \hat{I}\,\hat{h}$

- Input: image I, mask h
- `H=zeros(size(I))`, put h in the middle of H
- `H=ifftshift(H)`
- `FI=fft2(I) ; FH=fft2(H)`
- `FJ=FI.*FH`
- `J=real(ifft2(FJ))`

Matlab's `fftshift` and `ifftshift`

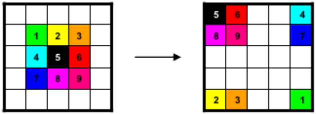
`J = fftshift(I):`

$$I(1,1) \rightarrow J(\lfloor R/2 \rfloor + 1, \lfloor C/2 \rfloor + 1)$$



`I = ifftshift(J):`

$$J(\lfloor R/2 \rfloor + 1, \lfloor C/2 \rfloor + 1) \rightarrow I(1,1)$$



where $\lfloor x \rfloor$ = `floor(x)` = the largest integer smaller than x.

Filters

Low pass

- pixel averaging, a weighted average of neighbors
- convolution with $\sum h = 1$
- high frequencies suppressed
- $\hat{h}$ decreases with $|f|$

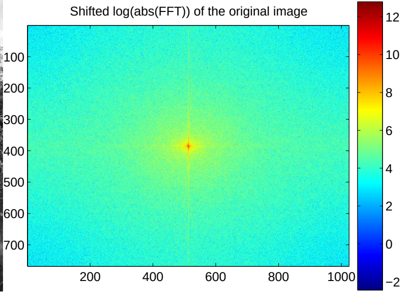
High pass

- pixel differences, a difference between neighbors
- convolution with $\sum h = 0$
- low frequencies suppressed
- $\hat{h}$ increases with $|f|$

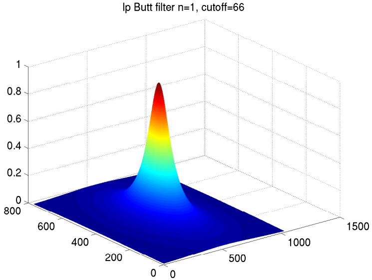
$\hat{h}_{\mathrm{HP}}(f) = 1 - \hat{h}_{\mathrm{LP}}(f)$

Filter shapes — ideal, Gaussian, Butterworth, Chebyshev, Bessel...

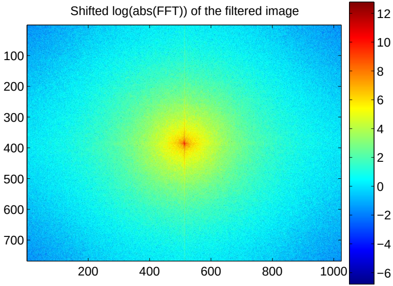
Lowpass filtering — Butterworth filter I



Lowpass filtering — Butterworth filter II



Butterworth lowpass filter



FFT of the filtered image

Maximally flat passband filter

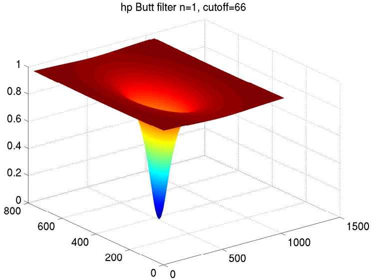
$H_{lp}(u,v) = \frac{1}{1+(D(u,v)/D_0)^{2/n}}$, where $D(u,v) = \sqrt{u^2+v^2}$

Lowpass filtering — Butterworth filter III

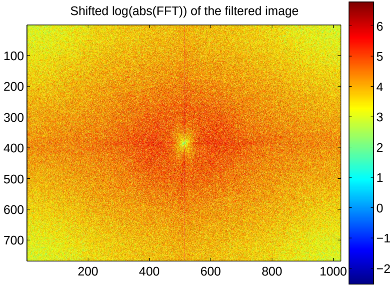
Original image

Filtered image

Highpass filtering — Butterworth filter I



Butterworth highpass filter

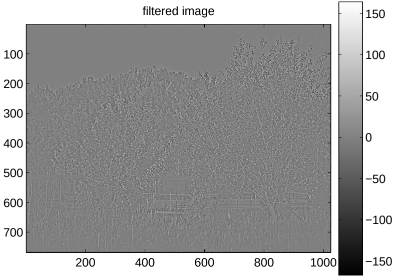


FFT of the filtered image

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

Highpass filtering — Butterworth filter II

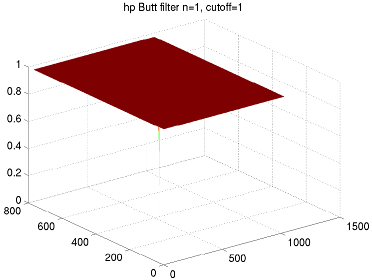
Original image



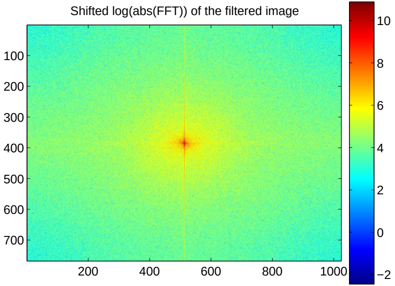
Filtered image

DC component lost, some values are negative.

Highpass filtering — Narrow filter



Butterworth highpass filter

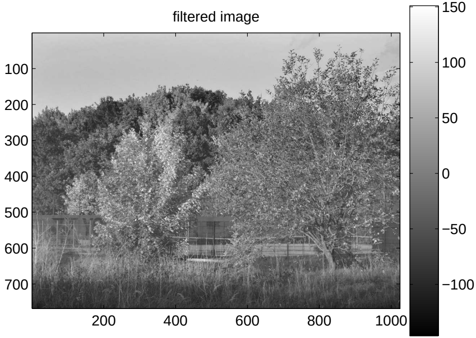


FFT of the filtered image

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

Highpass filtering — Narrow filter II

Original image

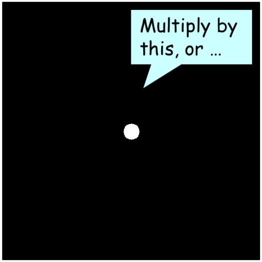


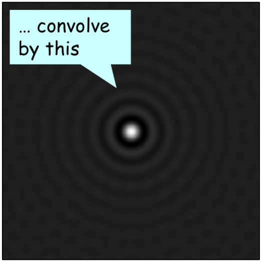
Filtered image

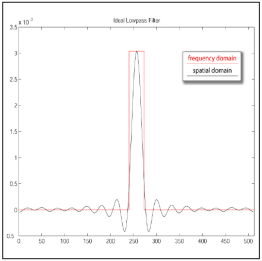
A very gentle high-pass filter. Original image is recovered except the DC component.

Ideal Lowpass Filter

Image size: 512x512
FD filter radius: 16







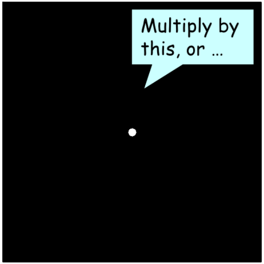
Fourier Domain Rep.

Spatial Representation

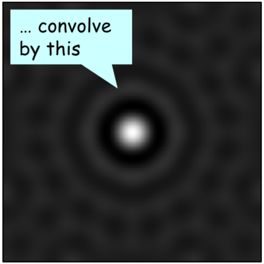
Central Profile

Ideal Lowpass Filter

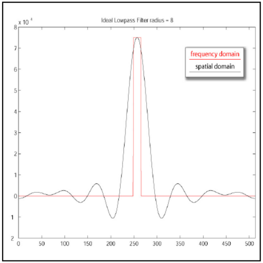
Image size: 512x512
FD filter radius: 8



Fourier Domain Rep.

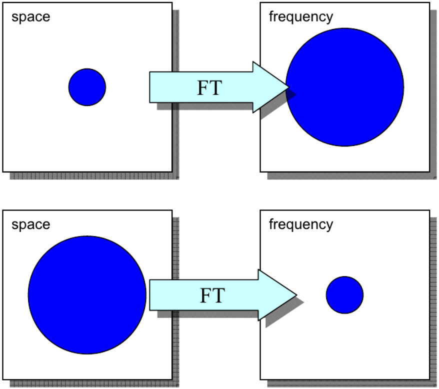


Spatial Representation



Central Profile

The Uncertainty Relation



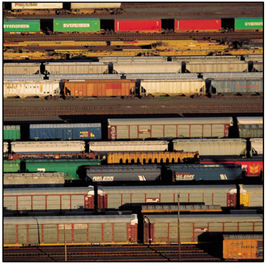
If $\Delta x \, \Delta y$ is the extent of the object in space and if $\Delta u \, \Delta v$ is its extent in frequency then,

$$\Delta x \, \Delta y \cdot \Delta u \, \Delta v \geq \frac{1}{16\pi^2}$$

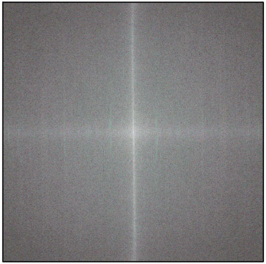
A small object in space has a large frequency extent and vice-versa.

Ideal Lowpass Filter

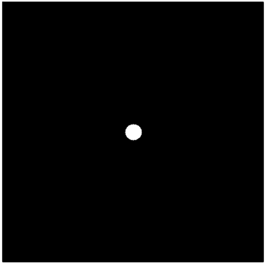
Image size: 512x512
FD filter radius: 16



Original Image



Power Spectrum

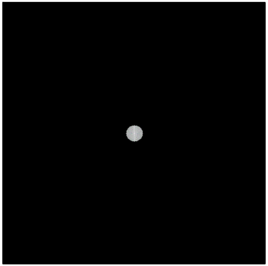


Ideal LPF in FD

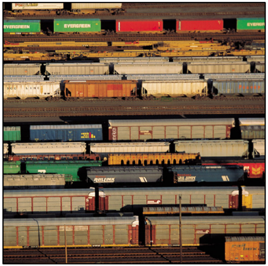
Ideal Lowpass Filter

Image size: 512x512
FD filter radius: 16

Filtered Image



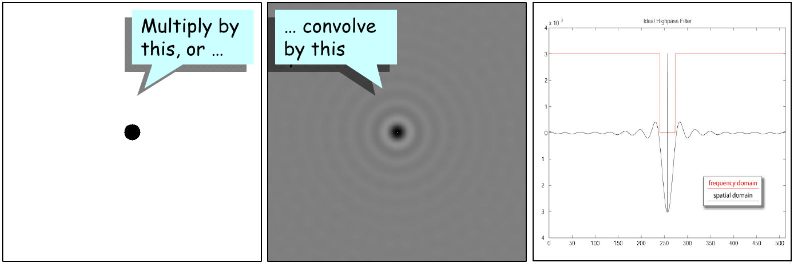
Filtered Power Spectrum



Original Image

Ideal Highpass Filter

Image size: 512x512
FD notch radius: 16



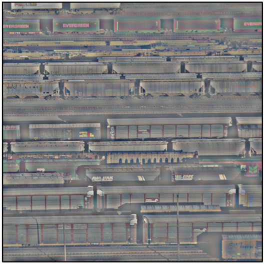
Fourier Domain Rep.

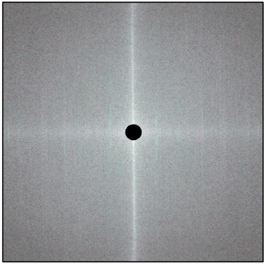
Spatial Representation

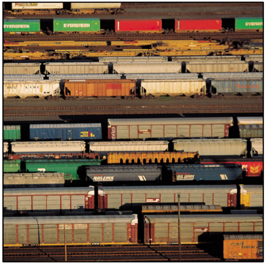
Central Profile

Ideal Highpass Filter

Image size: 512x512
FD notch radius: 16





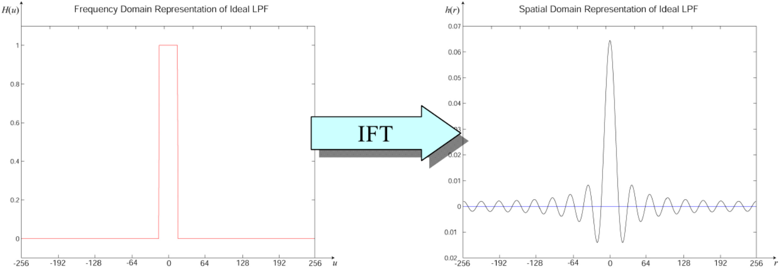


Filtered Image*

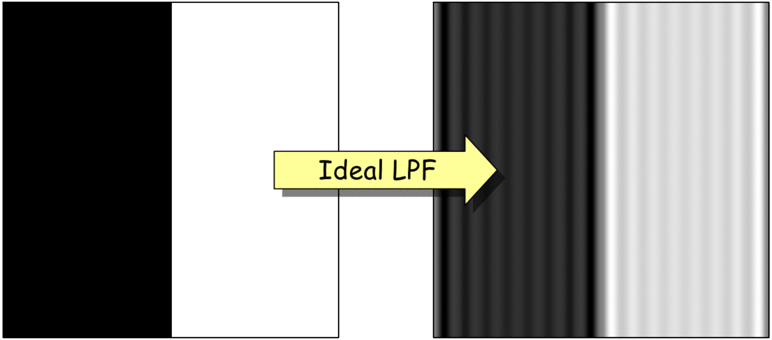
Filtered Power Spectrum

Original Image

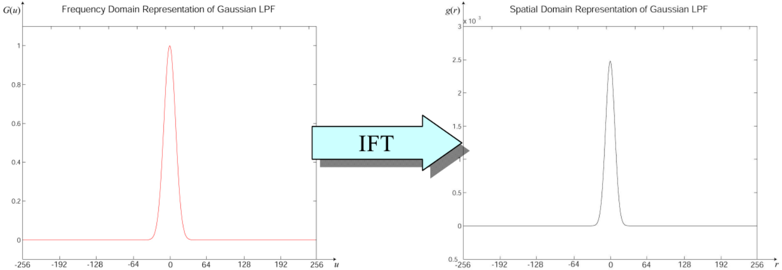
Ringing artifacts



Ringing artifacts

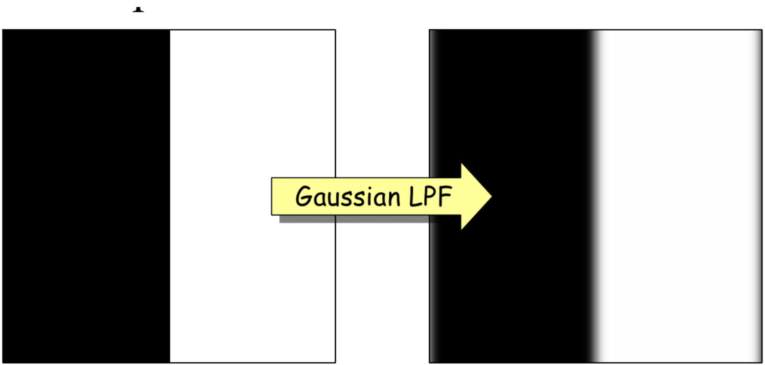


Gaussian filter



FT of a Gaussian is a Gaussian

Gaussian filter

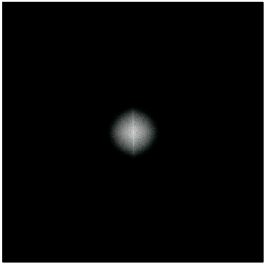


Blurring but no ringing

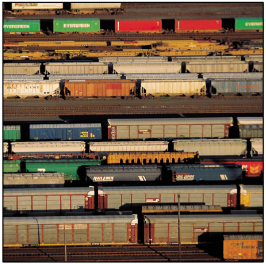
Gaussian Lowpass Filter

Image size: 512x512
SD filter sigma = 8

Filtered Image



Filtered Power Spectrum

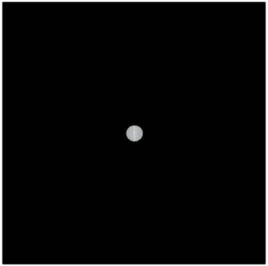


Original Image

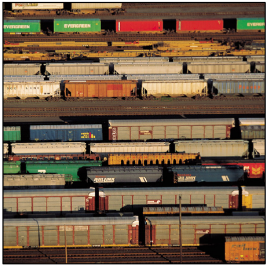
Ideal Lowpass Filter

Image size: 512x512
FD filter radius: 16

Filtered Image



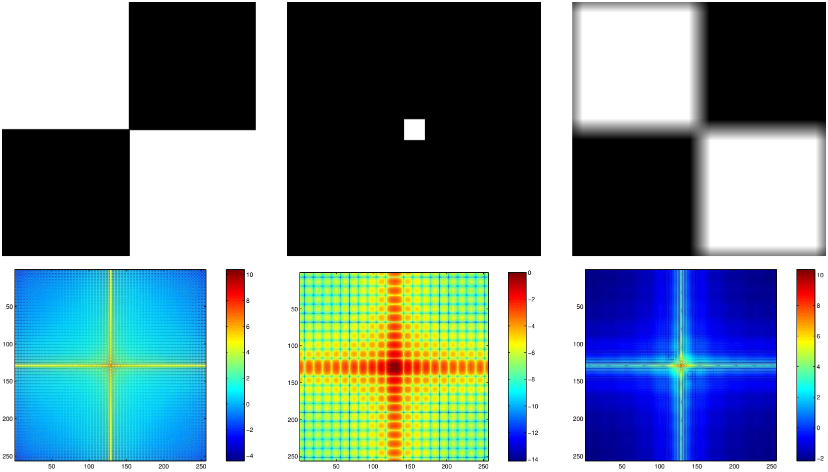
Filtered Power Spectrum



Original Image

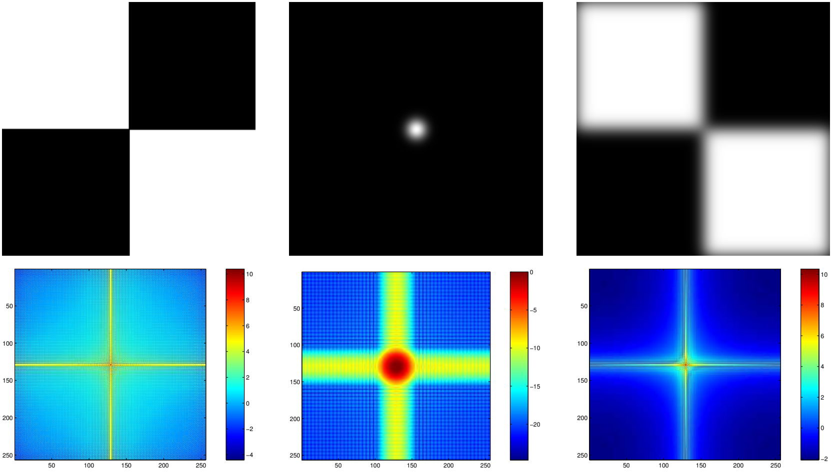
Frequency analysis of the spatial convolution — Simple averaging

Original image | 21×21 const. mask | filtered image



Frequency analysis of the spatial convolution — Gaussian smoothing

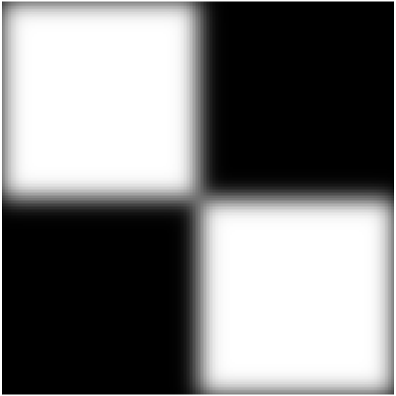
Original image | 21×21 Gauss. mask | filtered image



Simple averaging vs. Gaussian smoothing

simple averaging

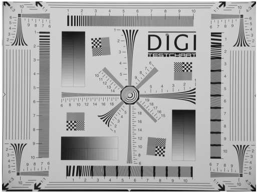
Gaussian smoothing



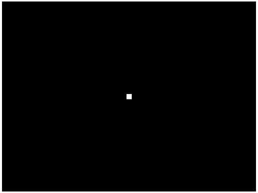
Both images blurred but filtering by a constant mask still shows up some high frequencies!

Frequency analysis of the spatial convolution — Simple averaging

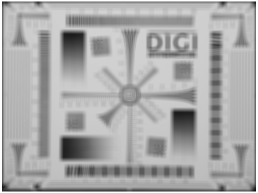
Original image

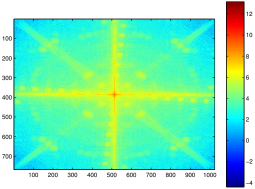


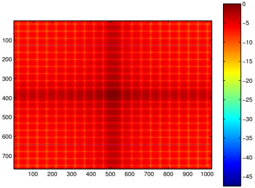
21×21 const. mask

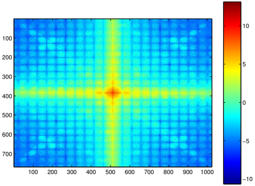


filtered image



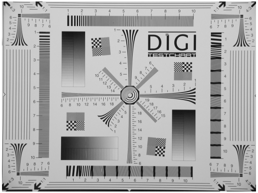


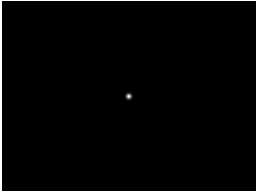


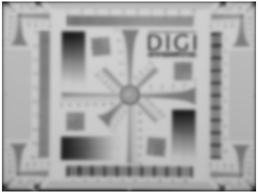


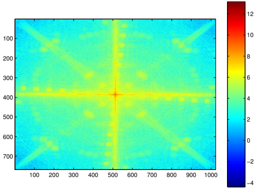
Frequency analysis of the spatial convolution — Gaussian smoothing

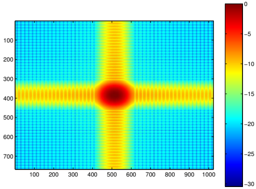
Original image | 21×21 Gauss. mask | filtered image

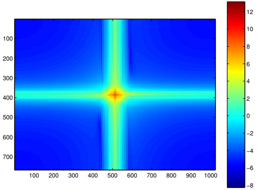






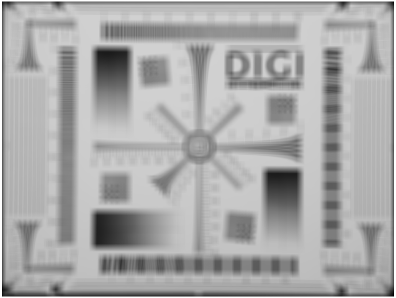




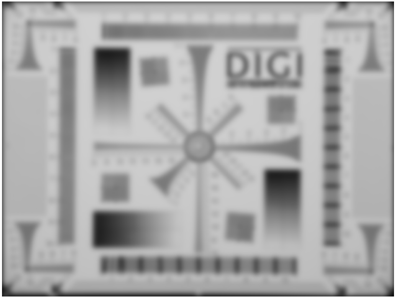


Simple averaging vs. Gaussian smoothing

simple averaging



Gaussian smoothing



Both images blurred but filtering by a constant mask still shows up some high frequencies!

Non-linear filtering

Typical goal: smooth homogeneous areas to reduce noise without blurring of image edges

Output is a **non-linear** function of a pixel value and those of its neighbours.

Simple linear averaging for noise reduction

3 × 3 neighborhood

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Center-weighted filters (approximate a Gaussian)

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Simple linear averaging for noise reduction

Example

original

Simple linear averaging for noise reduction

Example

with additive noise

Simple linear averaging for noise reduction

Example

filtered, 3×3 mask

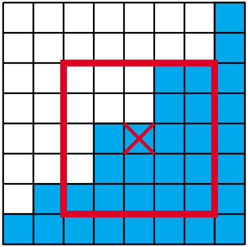
Simple linear averaging for noise reduction

Example

filtered, 7×7 mask

Non-linear smoothing

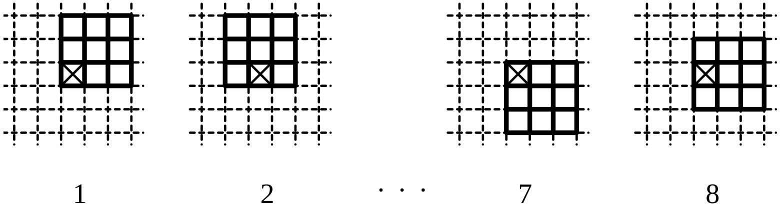
Goal: reduce blurring of image edges during smoothing
Homogeneous neighbourhood: find a proper neighbourhood where the values have minimal variance.



Robust statistics: something better than the mean.

Rotation mask

Rotation mask 3×3 seeks a homogeneous part at 5×5 neighbourhood.
Together 9 positions, 1 in the middle + 8 on the image



The mask with the lowest variance is selected and used for averaging.

Rotation mask—original image

Rotation mask—first filtration

Rotation mask—second filtration

Rotation mask—third filtration

Rotation mask—fourth filtration

Rotation mask—final (fifth) filtration

Nonlinear smoothing — Robust statistics

Order-statistic filters

- median
  - Sort values and **select** the middle one.
  - A method of **edge-preserving smoothing**.
  - Particularly useful for removing **salt-and-pepper**, or **impulse** noise.
- trimmed mean
  - Throw away outliers (e.g. 10% of the values) and average the rest.
  - More robust to a non-Gaussian noise than a standard averaging.

Median filtering

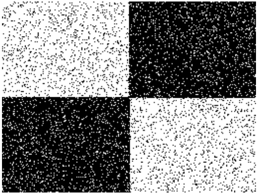
100	98	102
99	105	101
95	100	255

Mean = 117.2

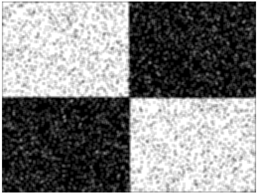
median: 95 98 99 100 100 101 102 105 255

Very robust, up to 50% of values may be outliers.

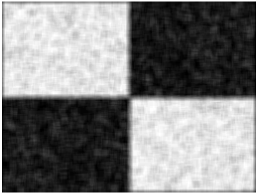
Nonlinear smoothing examples



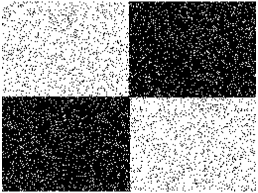
noisy image



averaging 3×3

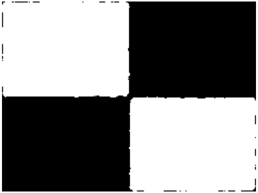


averaging 7×7



noisy image

median 3×3



median 7×7

- Suppresses impulse noise very well
- Damages thin edges

Median filtering for noise reduction

Example

original

Median filtering for noise reduction

Example

with additive noise

Median filtering for noise reduction

Example

median filtered, 3×3 mask

Filtering for object detection

Cross-correlation for pattern matching

$$g(x, y) = \sum_k \sum_l h(k, l) f(x + k, y + l) = h(x, y) \star f(x, y)$$

Cross-correlation is not, unlike convolution, commutative

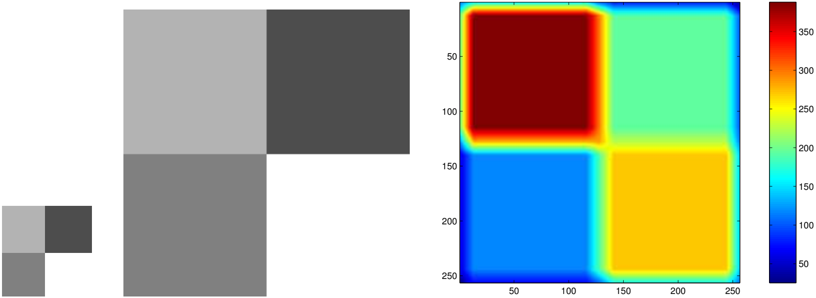
$$h(x, y) \star f(x, y) \neq f(x, y) \star h(x, y)$$

When $h(x, y) \star f(x, y)$ we often say that h **scans** f.
Cross-correlation is related to convolution through

$$h(x, y) \star f(x, y) = h(x, y) * f(-x, -y)$$

Cross-correlation is useful for **pattern matching**

Cross-correlation



$h(x, y)$ scans $f(x, y)$ $g(x, y)$

This is perhaps not exactly what we expected and what we want.
The result depends on the amplitudes.

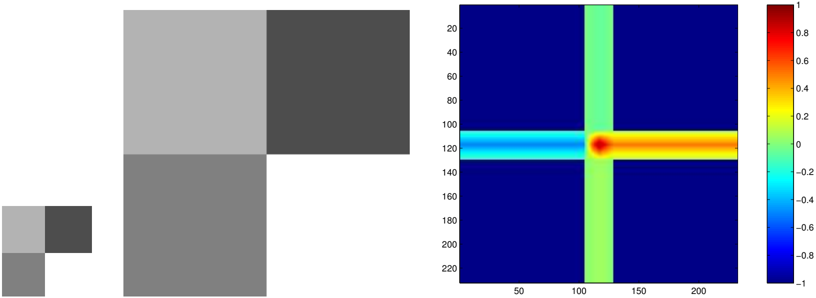
Normalised cross-correlation

Sometimes called **correlation coefficient**

$$c(x,y) = \frac{\sum_k \sum_l \left(h(k,l) - \overline{h}\right)\left(f(x+k,y+l) - \overline{f(x,y)}\right)}{\sqrt{\sum_k \sum_l \left(h(k,l) - \overline{h}\right)^2 \sum_k \sum_l \left(f(x+k,y+l) - \overline{f(x,y)}\right)^2}}$$

- $\overline{h}$ is the mean of h
- $\overline{f(x,y)}$ is the mean of the neighbourhood around (x,y)
- $\sum_k \sum_l \left(h(k,l) - \overline{h}\right)^2$ and $\sum_k \sum_l \left(f(x+k,y+l) - \overline{f(x,y)}\right)^2$ are variances.
- $-1 \leq c(x,y) \leq 1$

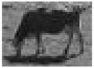
Normalised cross-correlation



$h(x, y)$ $f(x, y)$ $g(x, y)$

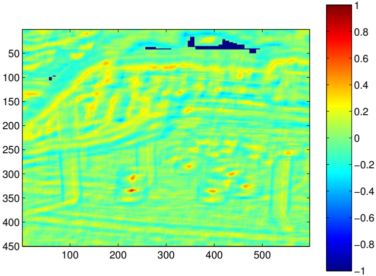
The dark blue regions stand for undefined values (NaN), where variance is zero.

Normalised cross-correlation — real images



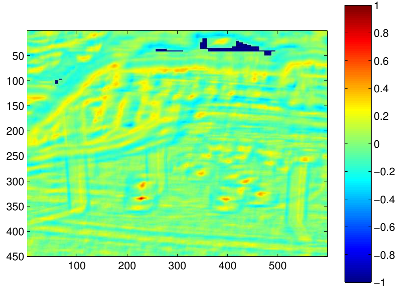
$h(x, y)$

$f(x, y)$



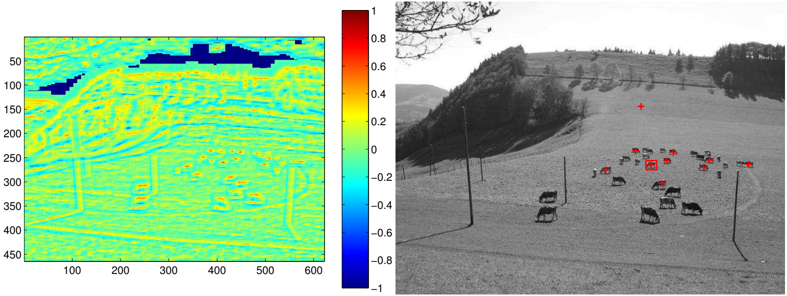
$g(x, y)$

Normalised cross-correlation — non-maxima suppression



Red rectangle denotes the **pattern**. The crosses are the 5 highest values of NCC after non-maxima suppression.

Normalised cross-correlation — non-maxima suppression



Red rectangle denotes the **pattern**. The crosses are the 10 highest values of NCC after non-maxima suppression.
The algorithm finds the cow in any position in the image.
(However, it does not scale)

Filtering for visual image improvement

- Local contrast adjustment
- Practical sharpening

Homomorphic filtering

- Aim: normalize the brightness across an image; increase contrast.
- Image is a product of illumination and reflectance components: $f(x, y) = i(x, y)r(x, y)$
- Illumination i — slow spatial variations (low frequency)
- Reflectance r — fast varitations (dissimilar objects)
- Use logarithm to separate the components
- Filter the logarithms

Homomorphic filtering — cont.

$$f(x, y) = i(x, y)r(x, y)$$

$$z(x, y) = \ln f(x, y) = \ln i(x, y) + \ln r(x, y)$$

Fourier pair

$$Z(u, v) = I(u, v) + R(u, v)$$

Filtering by a high-pass filter

$$S(u, v) = H(u, v)Z(u, v) = H(u, v)I(u, v) + H(u, v)R(u, v)$$

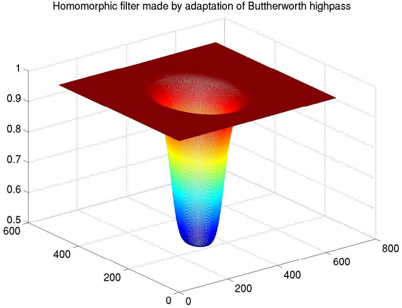
back to space $s(x, y) = \mathcal{F}^{-1}\{S(u, v)\}$ and back from log domain

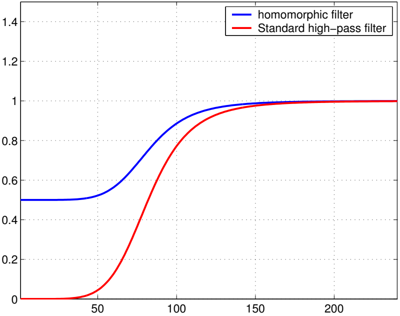
$$g(x, y) = \exp\left(s(x, y)\right)$$

We suppress variations in illumination and enhance reflectance component.

Homomorphic filtering — filters

Modified Butterworth filter





Remember: The filter is applied to $Z(u, v)$. Not to $F(u, v)$!

Homomorphic filtering — results

Original image.

Filtered image.

Image sharpening

- Correct for optical imperfection
- Correct for incorrect focus
- Improve visual appearance

Laplace filter sharpening

$$\nabla^2 f = \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 f \approx \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} * f + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} * f$$

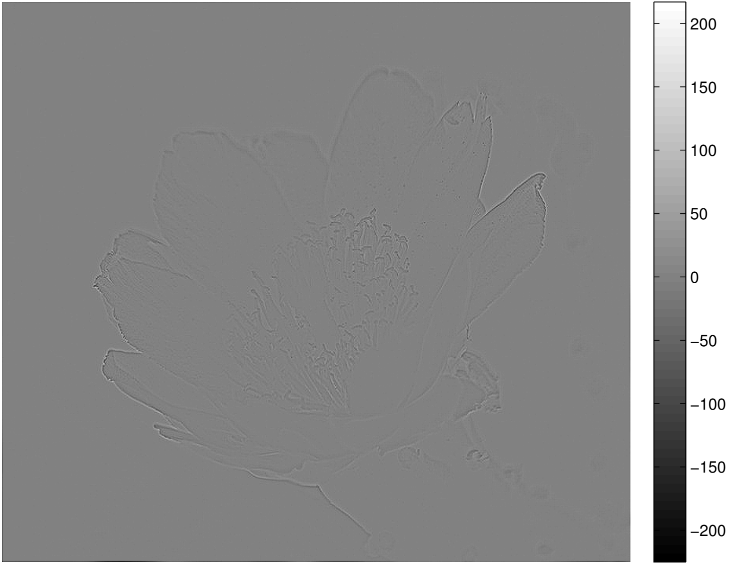
$$= \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{h} * f$$

Other approximations for $h \approx \nabla^2$ possible.

Laplace filter sharpening

original in black&white

Laplace filter sharpening



Laplace filtered

Laplace filter sharpening

$$\check{f} = f - w\nabla^2 f$$

original

Laplace filter sharpening

$$\check{f} = f - w\nabla^2 f$$

sharpened, $w = 1$, clipped to the original range

Laplace filter sharpening

$$(R, G, B) \to (H, S, V) \to (\check{H}, S, V) \to (R, G, B)$$

original

Laplace filter sharpening

$(R, G, B) \to (H, S, V) \to (\check{H}, S, V) \to (R, G, B)$

sharpened, $w = 1$

Unsharp masking

$$f_e = f - (f * G_\sigma) \qquad \check{f} = f + \alpha f_e = (1+\alpha)f - \alpha(f * G_\sigma)$$

original

Unsharp masking

original in black&white

Unsharp masking

smoothed, $\sigma = 3$

Unsharp masking

sharpened, $\alpha = 0.9$

Unsharp masking

original

Unsharp masking

sharpened, $\alpha = 0.9$

Unsharp masking II

Parameter dependence

original in black&white

Unsharp masking II

Parameter dependence

unsharp masked, $\sigma = 3$, $\alpha = 0.9$

Unsharp masking II

Parameter dependence

unsharp masked, $\sigma = 15$, $\alpha = 0.6$

Unsharp masking II

Parameter dependence

original

Unsharp masking II

Parameter dependence

unsharp masked, $\sigma = 3$, $\alpha = 0.9$

Unsharp masking II

Parameter dependence

unsharp masked, $\sigma = 15$, $\alpha = 0.6$

Unsharp masking III

Reducing noise sensitivity

$$
\begin{aligned}
f_e &= f - (f * G_\sigma) \\
\check{f} &= f + \alpha f_e = (1+\alpha)f - \alpha(f * G_\sigma)
\end{aligned}
$$

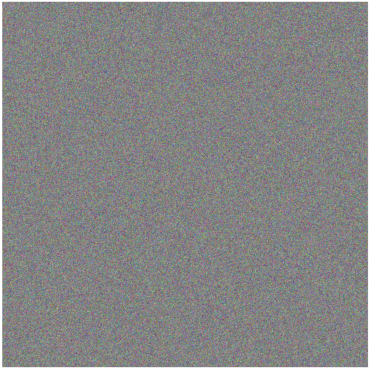
Unsharp masking III

Reducing noise sensitivity

$$
\begin{aligned}
f_e &= f - (f * G_\sigma) \\
\check{f} &= \begin{cases} (1+\alpha)f - \alpha(f * G_\sigma) & \text{if} |\nabla f| > T \\ f & \text{otherwise} \end{cases}
\end{aligned}
$$

Noise-Free Image and Uncorrelated Noise Field

image



Gaussian noise field

Spectra of Noise-Free Image and Uncorr. Noise Field

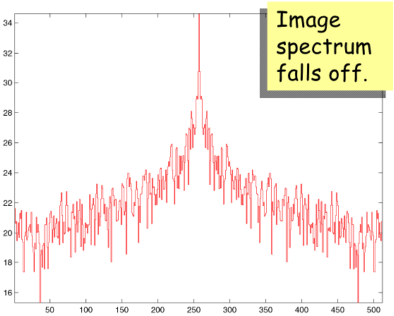
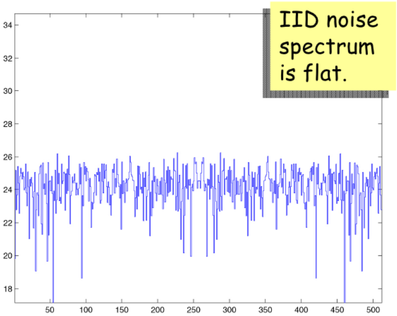


image center row log power spectrum



noise field center row log power spectrum

Sum of Noise-Free Image and Uncorrelated Noise Field

image + noise field

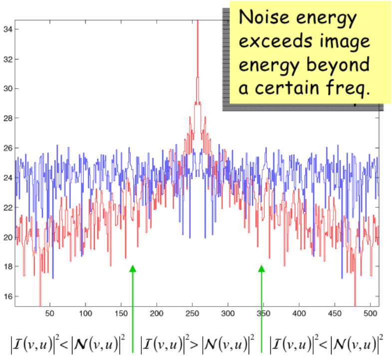
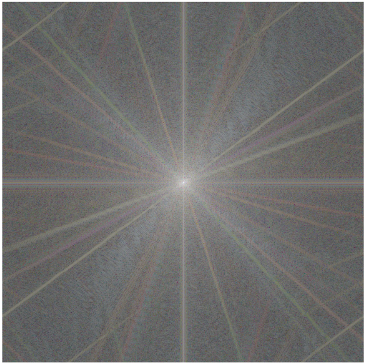
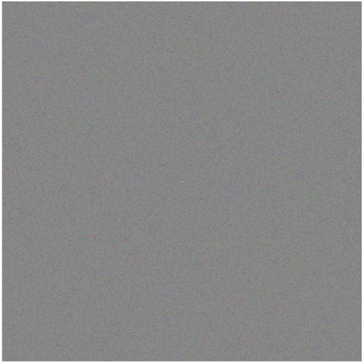


image + noise field center row log PS

Power Spectra of Noise-Free Image and Noise Field

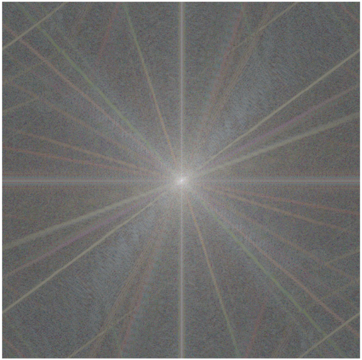


original image

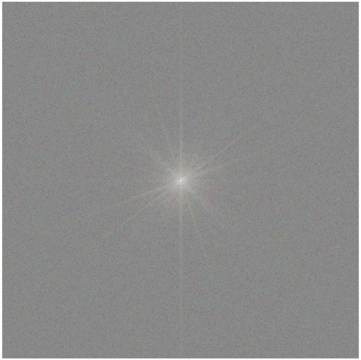


noise image

Power Spectra of Sum of Image and Noise Field

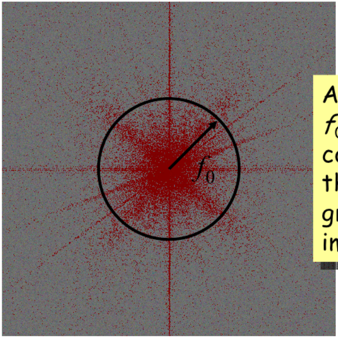


original image

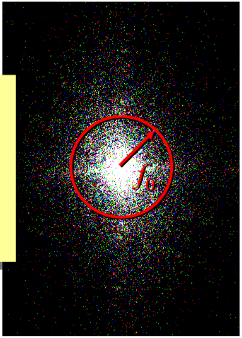


noisy image

Additive Noise: Reduce Through Blurring?



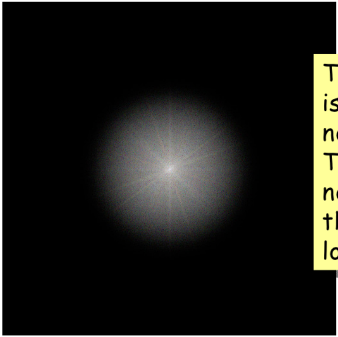
At some frequency, f_0, there are more components where the noise power is greater than the image power.



red indicates image > noise

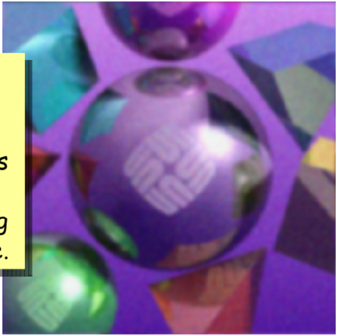
image PS > noise PS

Additive Noise: Reduction Through Blurring.



PS of Gaussian blurred image

The result is actually no better. There's less noise but the blurring looks worse.



Gaussian Blurred Image

Image Degradation Model

So far, we have considered only additive noise. Before going further it will be useful to consider a more general model of image degradation, one that includes convolution with a pointspread[1] function, H, as well as additive noise.

$$\overbrace{\mathrm{J}(r,c)}^{\text{degraded image}} = \underbrace{\mathrm{I}(r,c)}_{\text{undegraded image}} * \overbrace{\mathrm{H}(r,c)}^{\text{pointspread function}} + \underbrace{\mathrm{N}(r,c)}_{\text{additive noise}}.$$

[1]H is also referred to as the optical transfer function.

Lenses

A properly designed lens will focus the light emanating from a point and thereby reduce the blurring. But no lens can do this perfectly. In fact, the lens adds its own distortion. The result is an optical transfer function, H(r,c), that is convolved with the image.

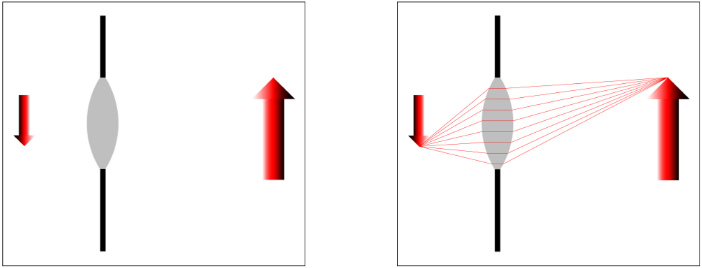
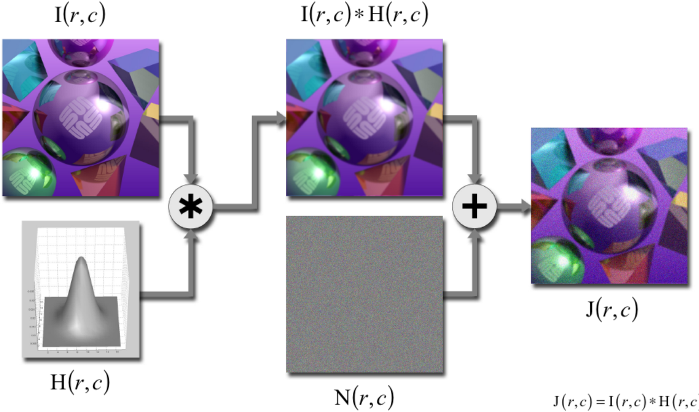


Image Degradation Model

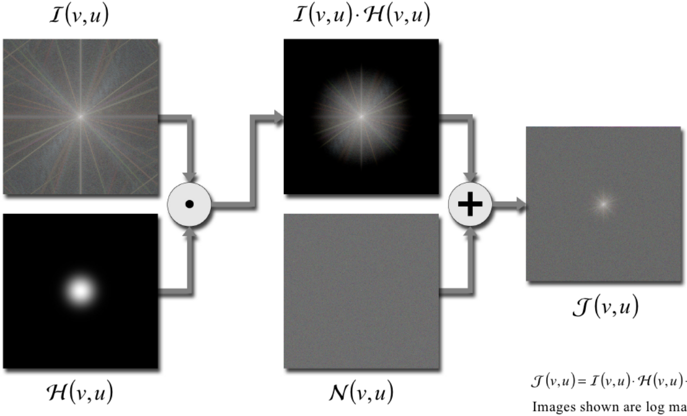


$$J(r,c) = I(r,c) * H(r,c) + N(r,c)$$

Image Degradation Model (Frequency Domain)

$$\underbrace{\mathcal{J}}_{\text{degraded image}}(v,u) = \underbrace{\mathcal{I}}_{\text{undegraded image}}(v,u) \cdot \underbrace{\mathcal{H}}_{\text{pointspread operator}}(v,u) + \underbrace{\mathcal{N}}_{\text{additive noise}}(v,u).$$

Image Degradation Model (Frequency Domain)



$\mathcal{J}(v,u) = \mathcal{I}(v,u) \cdot \mathcal{H}(v,u) + \mathcal{N}(v,u).$

Images shown are log magnitude.

Wiener filter

Problem definition

Observed image in frequency space

$$J(u,v) = I(u,v)H(u,v) + N(u,v) = I\,H + N$$

Find a filter

$$\hat{I}(u,v) = W(u,v)J(u,v) = W(IH + N)$$

$$\text{such that} \quad \varepsilon^2 = \mathrm{E}\left[\int |I(u,v) - \hat{I}(u,v)|^2 \,\mathrm{d}u\mathrm{d}v\right] \quad \text{is minimized}$$

Wiener filter

Derivation

$$\varepsilon^2 = \mathrm{E}\left[\int |I(u,v) - \tilde{I}(u,v)|^2 \,\mathrm{d}u\mathrm{d}v\right]$$

Minimize for each u, v

$$\mathrm{E}\left[|I(u,v) - \tilde{I}(u,v)|^2\right] = \mathrm{E}\left[|I - W(HI+N)|^2\right] =$$
$$|1 - WH|^2 \underbrace{\mathrm{E}\left[|I|^2\right]}_{P_I} + |W|^2 \underbrace{\mathrm{E}\left[|N|^2\right]}_{P_N} = |1 - WH|^2 P_I + |W|^2 P_N$$

since I and N are uncorrelated

Wiener filter

Derivation

$$\text{minimize} \quad |1 - WH|^2 P_I + |W|^2 P_N$$

Take a complex derivative with respect to W. Recall that $\left(|x|^2\right)' = \overline{x}$

$$\begin{aligned} -H\overline{1 - WH}P_I + \overline{W}P_N &= 0 \\ \overline{W}\left(|H|^2 P_I + P_N\right) &= HP_I \end{aligned}$$

The Wiener filter is

$$W = \frac{\overline{H}P_I}{|H|^2 P_I + P_N}$$

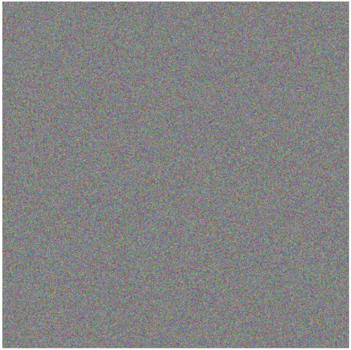
Wiener filter
Notes

$$W = \frac{\overline{H} P_I}{|H|^2 P_I + P_N}$$

- For frequencies where $P_I \gg P_N$, Wiener filter approximates an inverse filter, $W \approx 1/H$.
- For frequencies where $P_I \ll P_N$, Wiener filter filters the noise out, $W \approx 0$.
- Only P_I and P_N are needed. Sometimes $H = \delta$

Noise Reduction Through LMS Filtering[1]

image



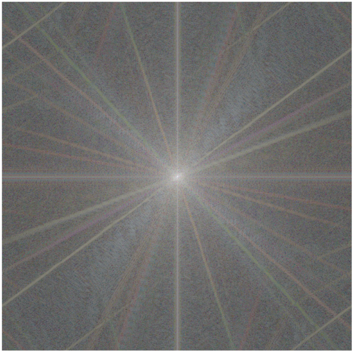
Gaussian noise field

Noise Reduction Through LMS Filtering[1]

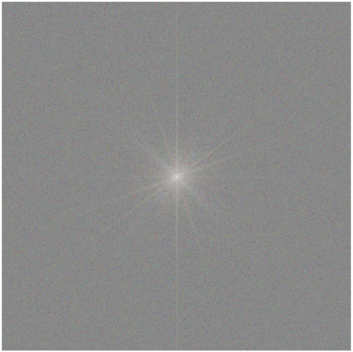
image

noisy image

Additive Noise (Power Spectra)



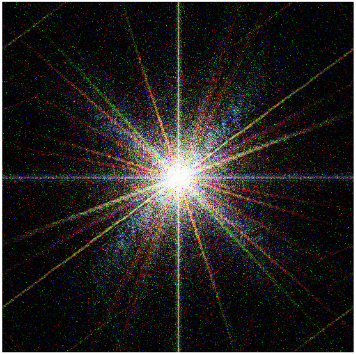
original image



noisy image

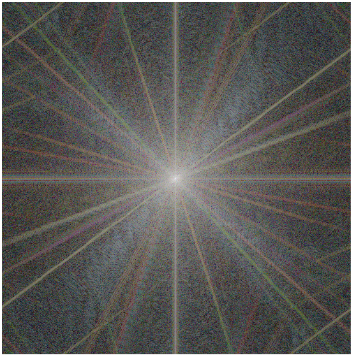
Additive Noise (Power Spectra)

Wiener filtered image

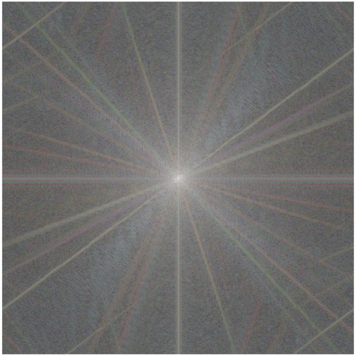


Wiener filter

Additive Noise (Power Spectra)



Wiener filtered image



original image

Additive Noise

noisy image

Wiener filtered image

Additive Noise

Wiener filtered image

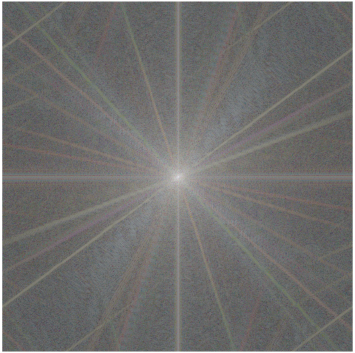
original image

Noise Reduction Through LMS Filtering[1]

image

noisy image $J = I*h + N$

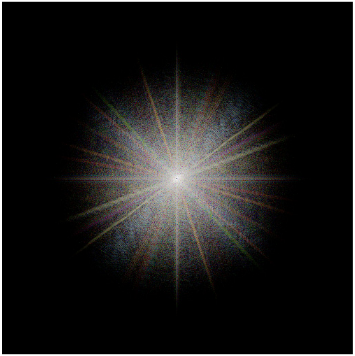
Image*PSF + Noise (Power Spectra)



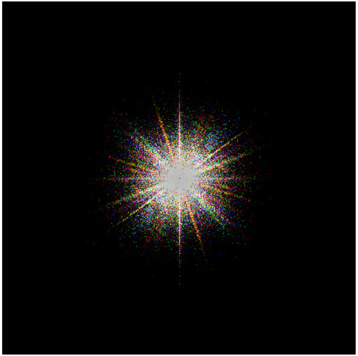
original image

noisy image $J = I*h + N$

Image*PSF + Noise (Power Spectra)



Wiener filtered image



Wiener filter

Image*PSF + Noise

noisy image $J = I*h + N$

Wiener filtered image

Image*PSF + Noise

Wiener filtered image

original image

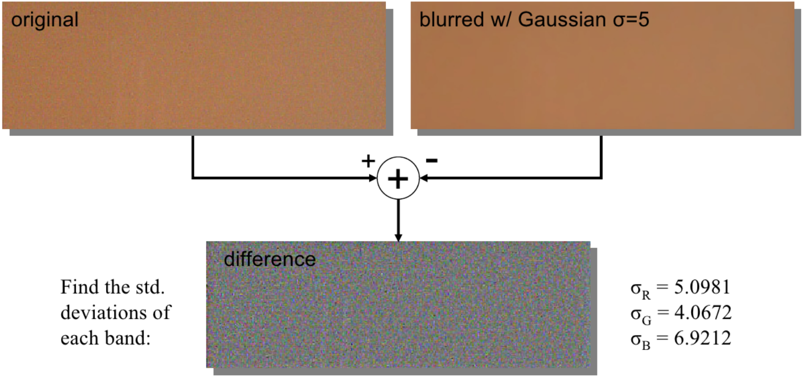
LMS Image Restoration (Real Example)

For this real example we need to estimate the image power spectrum, the pointspread function and the noise power spectrum.

LMS Image Restoration (Real Example)

To estimate the noise power spectrum, analyze a constant area from the image.

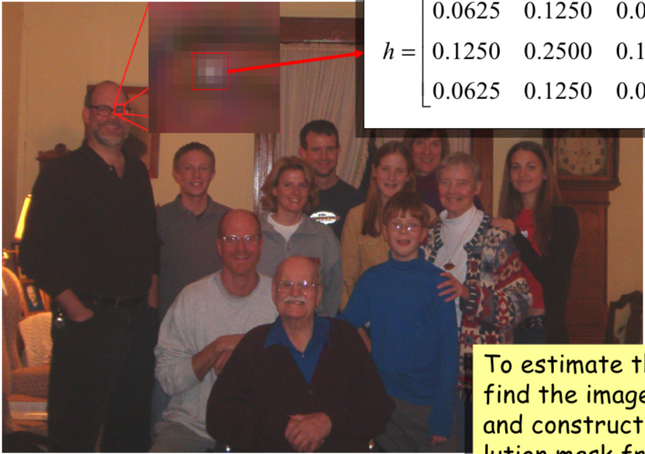
Noise Estimation



Find the std. deviations of each band:

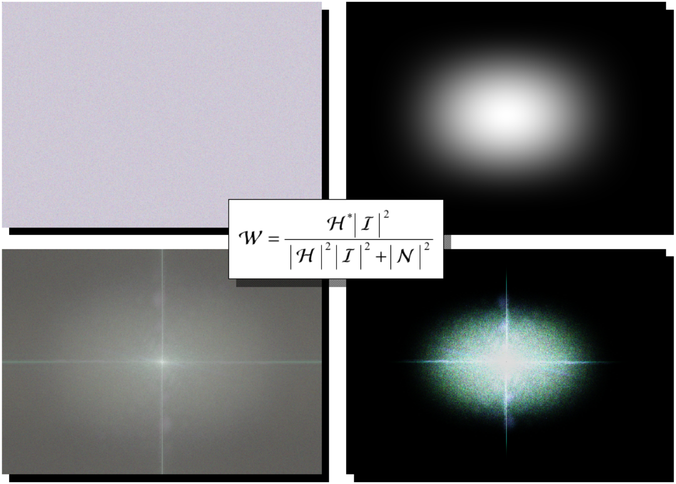
$\sigma_R = 5.0981$
$\sigma_G = 4.0672$
$\sigma_B = 6.9212$

Pointspread Function Estimation



$$h = \begin{bmatrix} 0.0625 & 0.1250 & 0.0625 \\ 0.1250 & 0.2500 & 0.1250 \\ 0.0625 & 0.1250 & 0.0625 \end{bmatrix}$$

To estimate the PSF, find the image of a point and construct a convolution mask from it.


$|\mathcal{N}|^2$
$|\mathcal{H}|^2$
$$\mathcal{W} = \frac{\mathcal{H}^* |\mathcal{I}|^2}{|\mathcal{H}|^2 |\mathcal{I}|^2 + |\mathcal{N}|^2}$$
$|\mathcal{I}|^2$
$\mathcal{W}$

LMS Image Restoration (filtered)

Detail of Results

The contrast of these has been increased to make the differences more visible.

original image

filtered image

matlab's wiener2

Local adaptive Wiener filtering

- Each window filtered separately.
- Signal variance estimated from image.
- Noise assumed i.i.d. Gaussian.
- Neglects spatial correlation
- Matlab function `wiener2`

Local adaptive Wiener filtering

original

Local adaptive Wiener filtering

original in black&white

Local adaptive Wiener filtering

original+noise, i.i.d. Gaussian, $\sigma = 60$

Local adaptive Wiener filtering

local Wiener filtering, `wiener2`

Local adaptive Wiener filtering

original in black&white

Introduction

Noise

Spatial filtering

Frequency domain filtering

Non-linear filtering

Filtering applications

Denoising by linear filtering

Wavelets

Wavelets

- Discrete wavelet transform

$$f(x) = \sum_k b_k \varphi(2^J x - k) + \sum_{j \geq J} \sum_m c_m^j \psi(2^j x - m)$$

φ — scaling function (low pass)
ψ — wavelet (high pass)
b — lowpass coefficients
c — highpass/wavelet/detail coefficients

  - Fast DWT algorithms

Wavelets

- Discrete wavelet transform

$$f(x) = \sum_k b_k \varphi(2^J x - k) + \sum_{j \geq J} \sum_m c_m^j \psi(2^j x - m)$$

φ — scaling function (low pass)
ψ — wavelet (high pass)
b — lowpass coefficients
c — highpass/wavelet/detail coefficients

  - Fast DWT algorithms

- 2D Discrete wavelet transform

$$f(x, y) = \sum_{k,k'} b_{k,k'} \varphi(2^J x - k)\varphi(2^J y - k') + \sum_{j \geq J} \sum_{m,m'} c_{m,m'}^l \psi(2^j x - m)\psi(2^j y - m')$$

  - Separability $\rightarrow$ quick decomposition

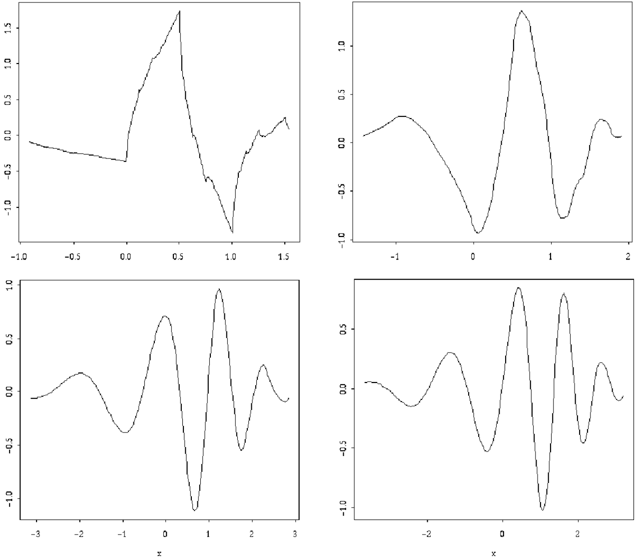
Wavelets

- Discrete wavelet transform
- Continuous wavelet transform
- Shift invariant (overcomplete) wavelet transform
- General decomposition

$$f(x,y) = \sum_{k,k'} b_{k,k'} \varphi_{k,k'}(x,y) + \sum_{j \geq J} \sum_{m,m'} c^j_m \psi_{m,m'}(x,y)$$

- Orthogonal basis functions, localized in space and frequency

Wavelets

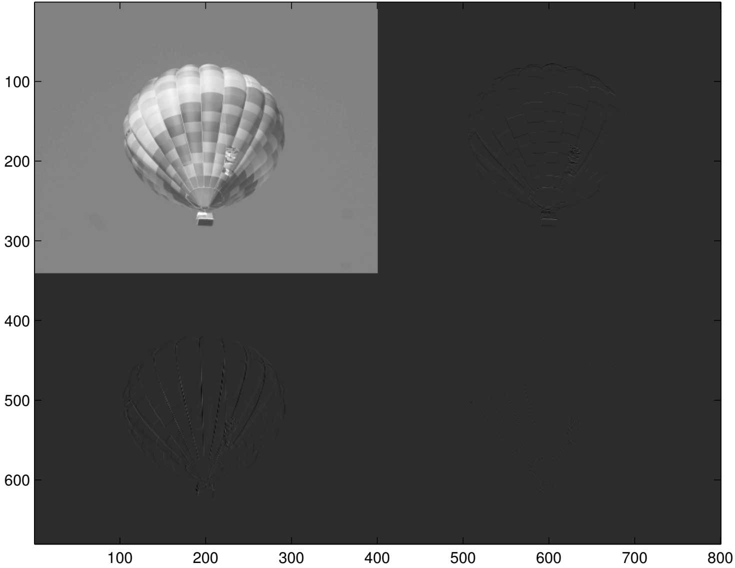


Daubechies family wavelets

Wavelet decomposition

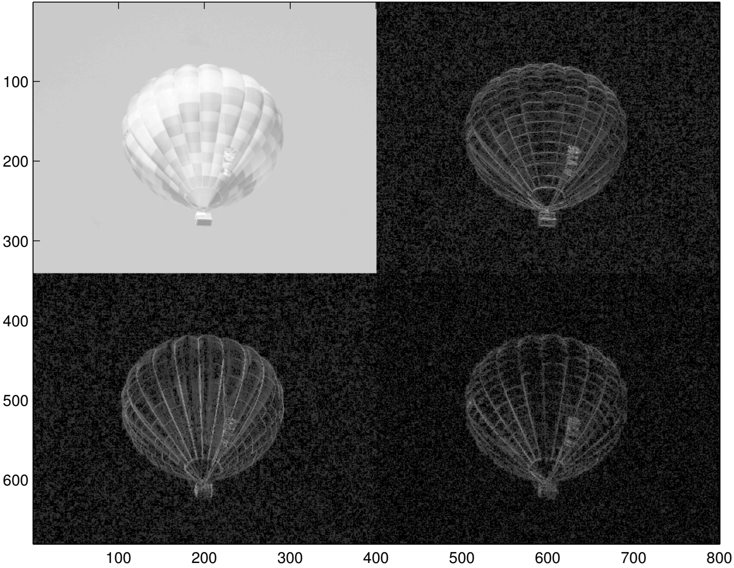
original in black&white

Wavelet decomposition



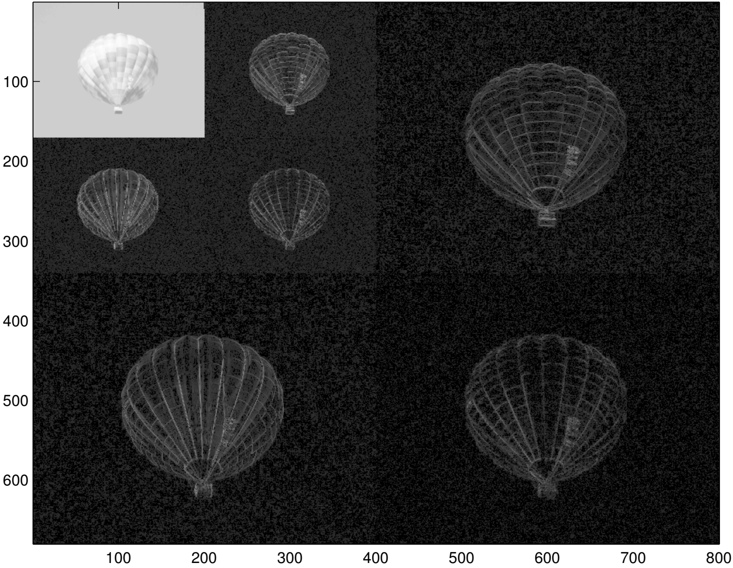
wavelet decomposition, 1 level, Haar, Matlab toolbox

Wavelet decomposition



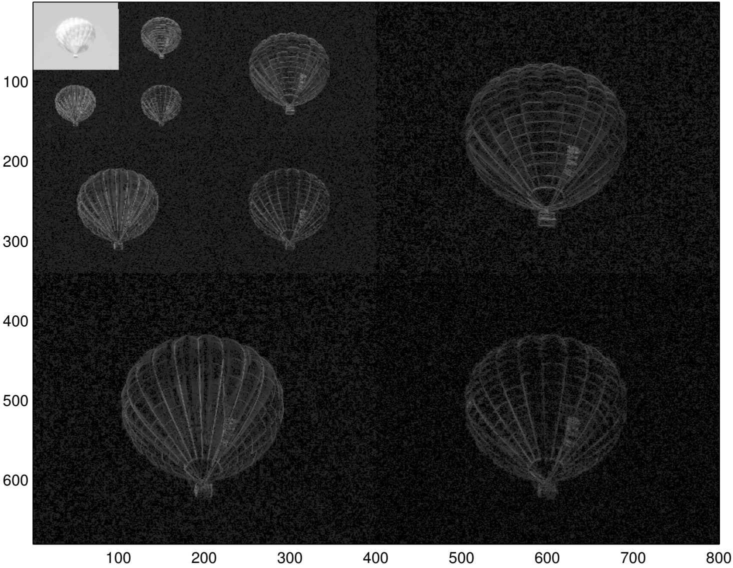
wavelet decomposition, 1 level, intensity rescaled

Wavelet decomposition



wavelet decomposition, 2 levels, intensity rescaled

Wavelet decomposition



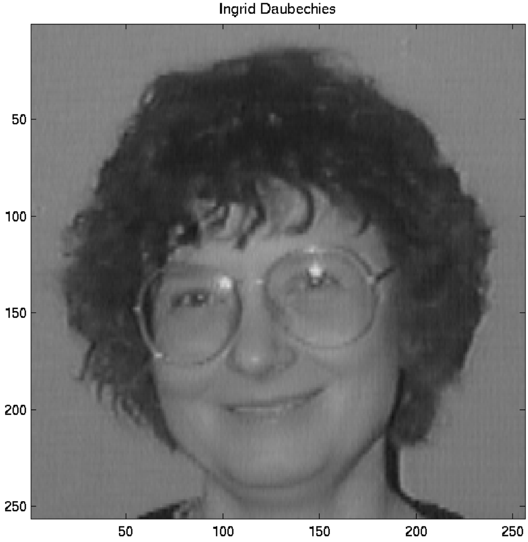
wavelet decomposition, 3 levels, intensity rescaled

Wavelet compression

- Wavelet transform (analysis)
- Order coefficients by magnitude
- Only use M largest (set the rest to zero)
- Inverse wavelet transform (synthesis)

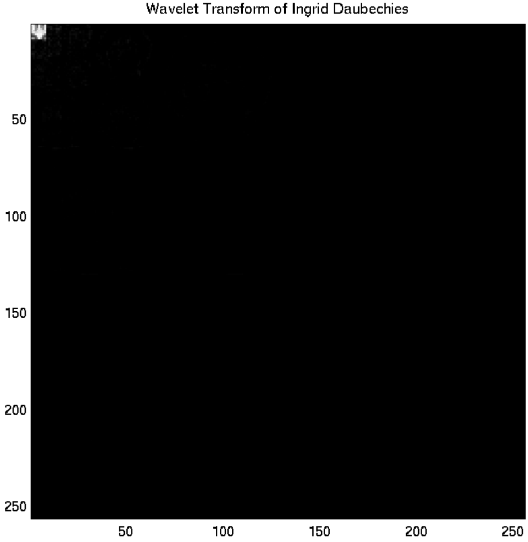
2D Wavelet compression example

Separable decomposition, alternate x and y.



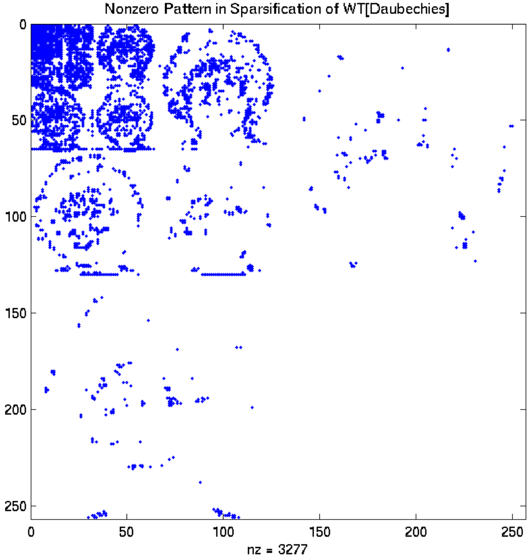
2D Wavelet compression example

Separable decomposition, alternate x and y.



2D Wavelet compression example

Separable decomposition, alternate x and y.



2D Wavelet compression example

Separable decomposition, alternate x and y.

2D Wavelet compression example

Separable decomposition, alternate x and y.

Wavelet denoising

Idea: small coefficients are due to noise

- Wavelet decomposition (DWT,SWT)
- Thresholding
- Wavelet reconstruction

Wavelet denoising

Idea: small coefficients are due to noise

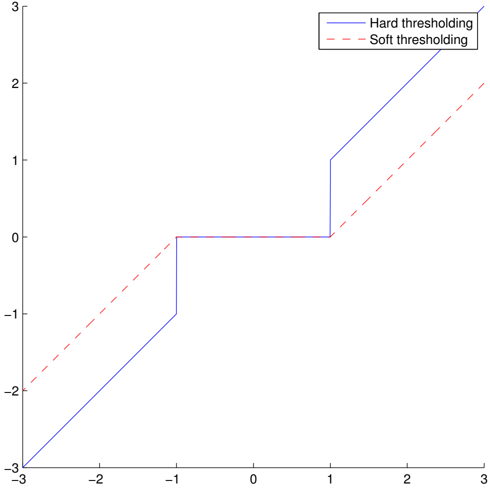
- Wavelet decomposition (DWT,SWT)
- Thresholding
- Wavelet reconstruction

Why does it work well:

- Wavelet decomposition is parsimonious (lot of zeros)
- Wavelets are orthogonal
- Wavelets are well localized in space *and* smooth
- Wavelets are multiscale
- Wavelet family is very large

Thresholding

- Hard and soft



Thresholding

- Hard and soft
- Threshold choice
  - Universal threshold (Donoho)

$$\lambda = \hat{\sigma}\sqrt{2 \log N}$$

  - $\hat{\sigma}$ is estimated from fine scale coefficients
  - SURE (Stein's unbiased estimator of risk), cross-validation...

Wavelet denoising example

original in black&white

Wavelet denoising example

noisy

Wavelet denoising example

wavelet denoised, symlets 4, threshold 10

Wavelet denoising example

wavelet denoised, symlets 4, threshold 20

Wavelet denoising example

wavelet denoised, symlets 4, threshold 40

Wavelet denoising example

wavelet denoised, symlets 4, threshold 60

Wavelet denoising example

wavelet denoised, symlets 4, threshold 80

Wavelet denoising example

wavelet denoised, symlets 4, threshold 100

Wavelet denoising example

wiener denoised

Conclusions

- Filtering in space
- Filtering in frequency domain
- Smoothing, sharpening
- Denoising
- Wiener filter
- Wavelet denoising