EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Computer Graphics (Graphische Datenverarbeitung)

- Image Processing -

WS 2021/2022

Corona

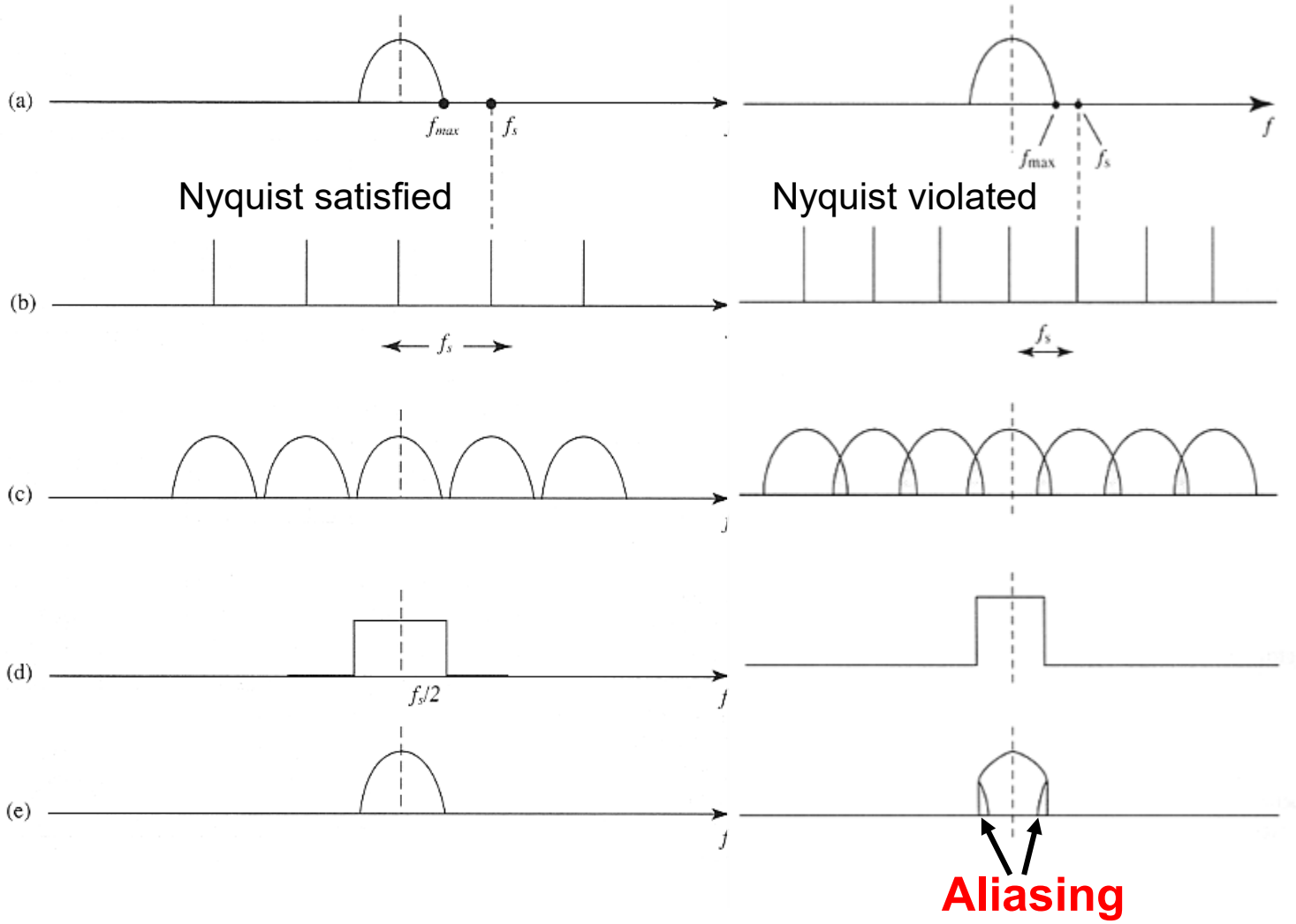
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Regular random lookup of the 3G certificates

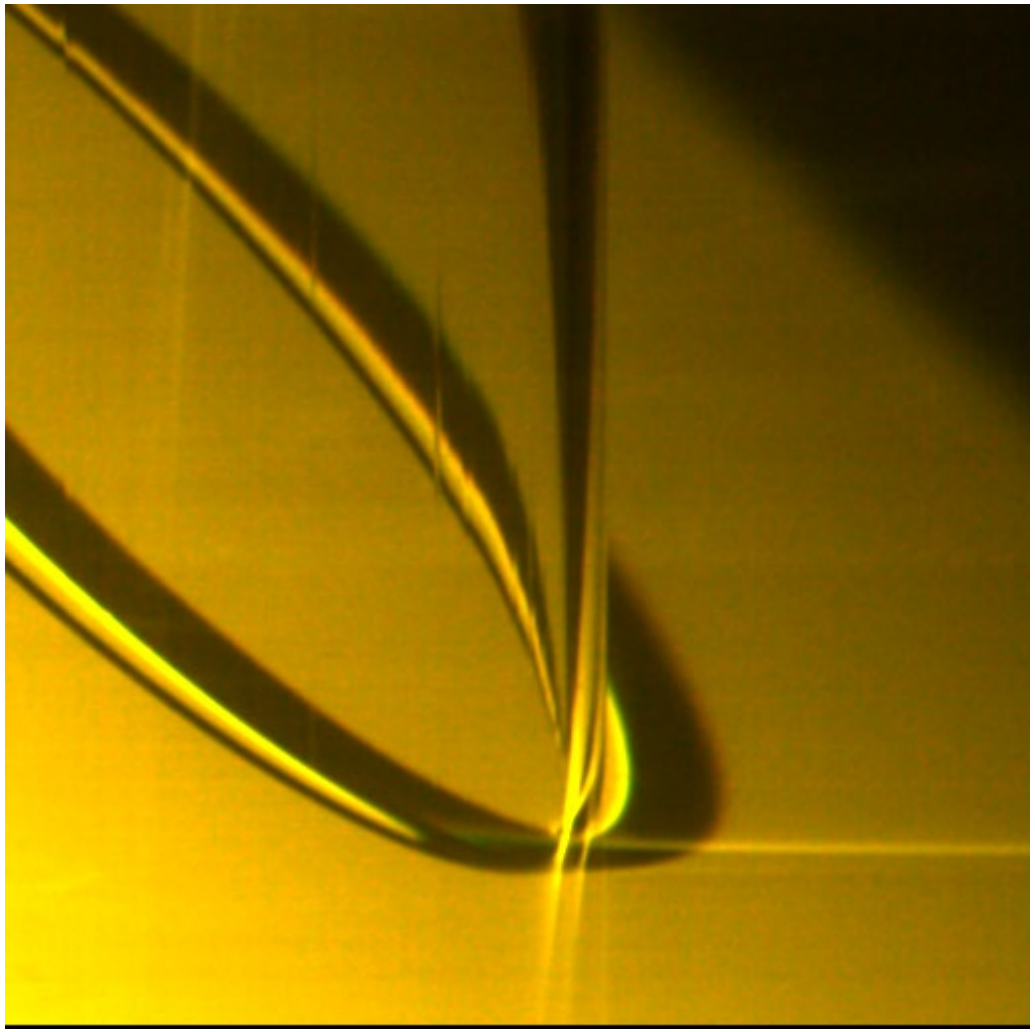
- Contact tracing: We need to know who is in the class room
  - New ILIAS group for every lecture slot
  - Register via ILIAS or this QR code (only if you are present in this room)

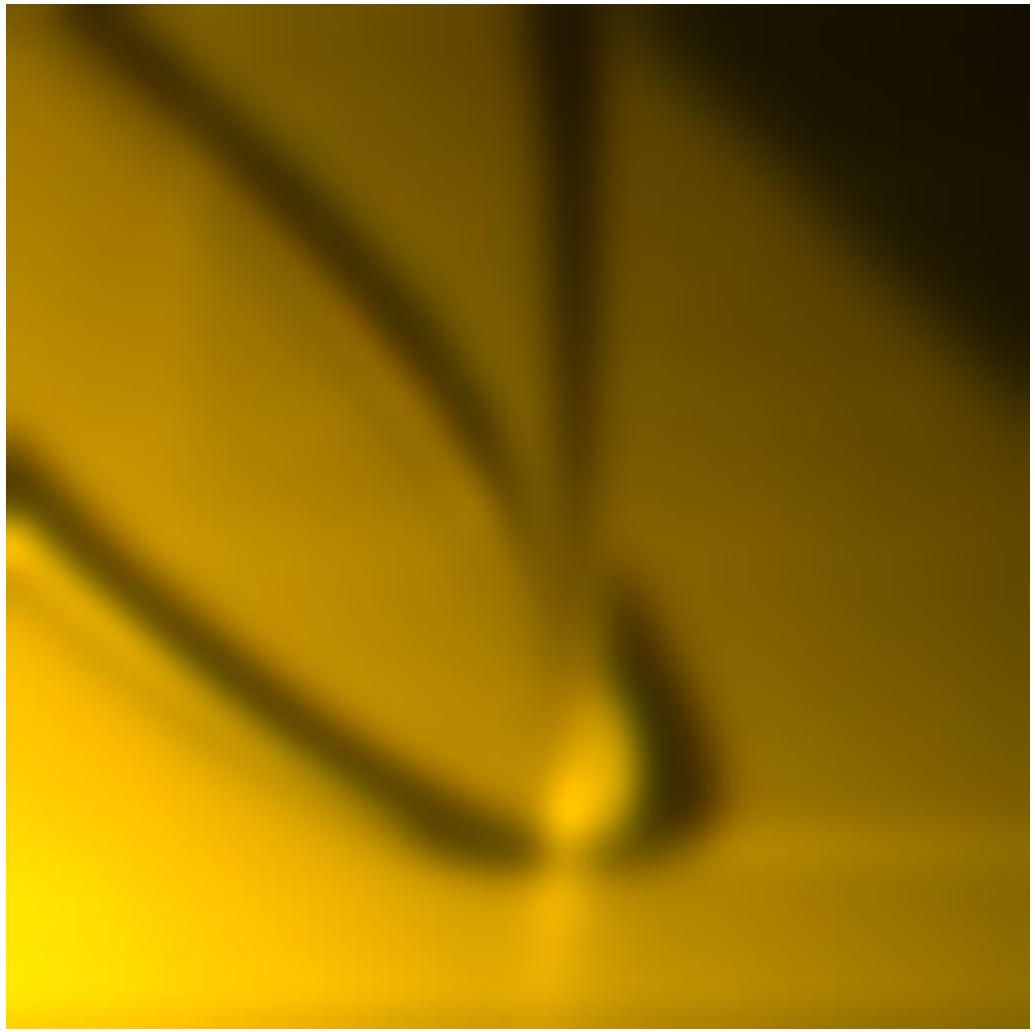
Aliasing

- In Fourier space
- Original spectrum
- Sampling comb
- Resulting spectrum
- Reconstruction Filter
- Reconstructed spectrum



What kind of filter operation?





Which Filters do you know?

Overview

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Last lecture
  - Sampling
  - Antialiasing & supersampling
- Now
  - Image Filters
  - Wavelets
- Next
  - Human Visual System

What you should learn today

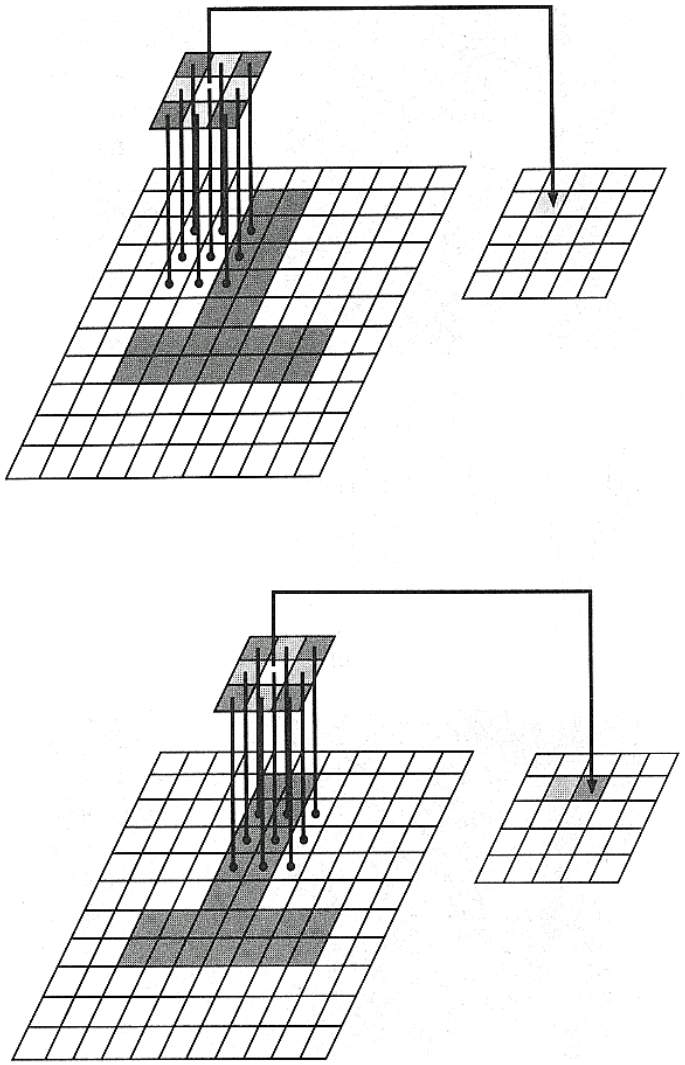
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Definition of standard filter kernels
- Properties of Wavelets
- Edge-Avoiding Wavelets
- When to apply which filter
- How to compute a filtered image

Image Filters

Convolution and Filtering

- Technical Realization
  - In image domain
  - Pixel mask with weights
  - OpenGL: Convolution extension
- Problems (e.g. sinc)
  - Large filter support
    - Large mask
    - A lot of computation
  - Negative weights
    - Negative light?



Gaussian Filter

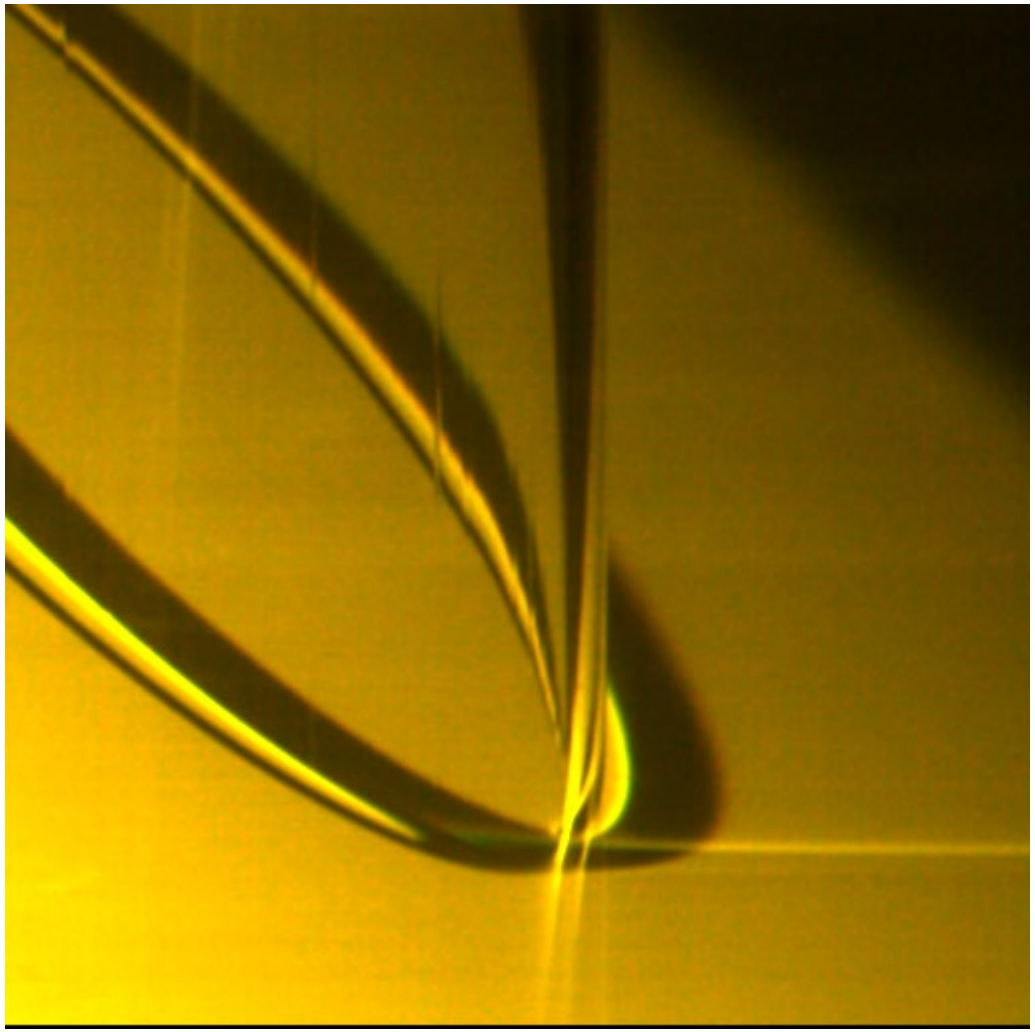
- Convolution with a 2D Gaussian

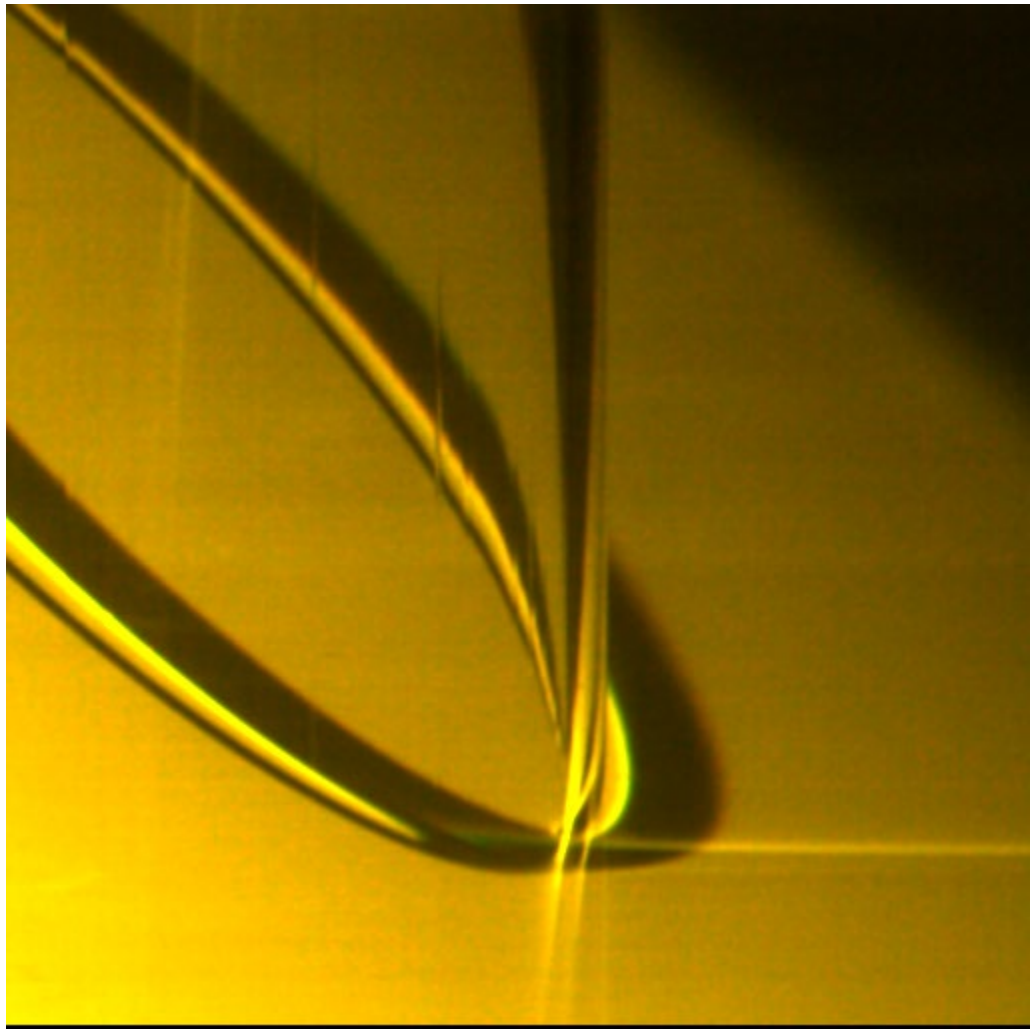
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- variance σ is defining the “blur”

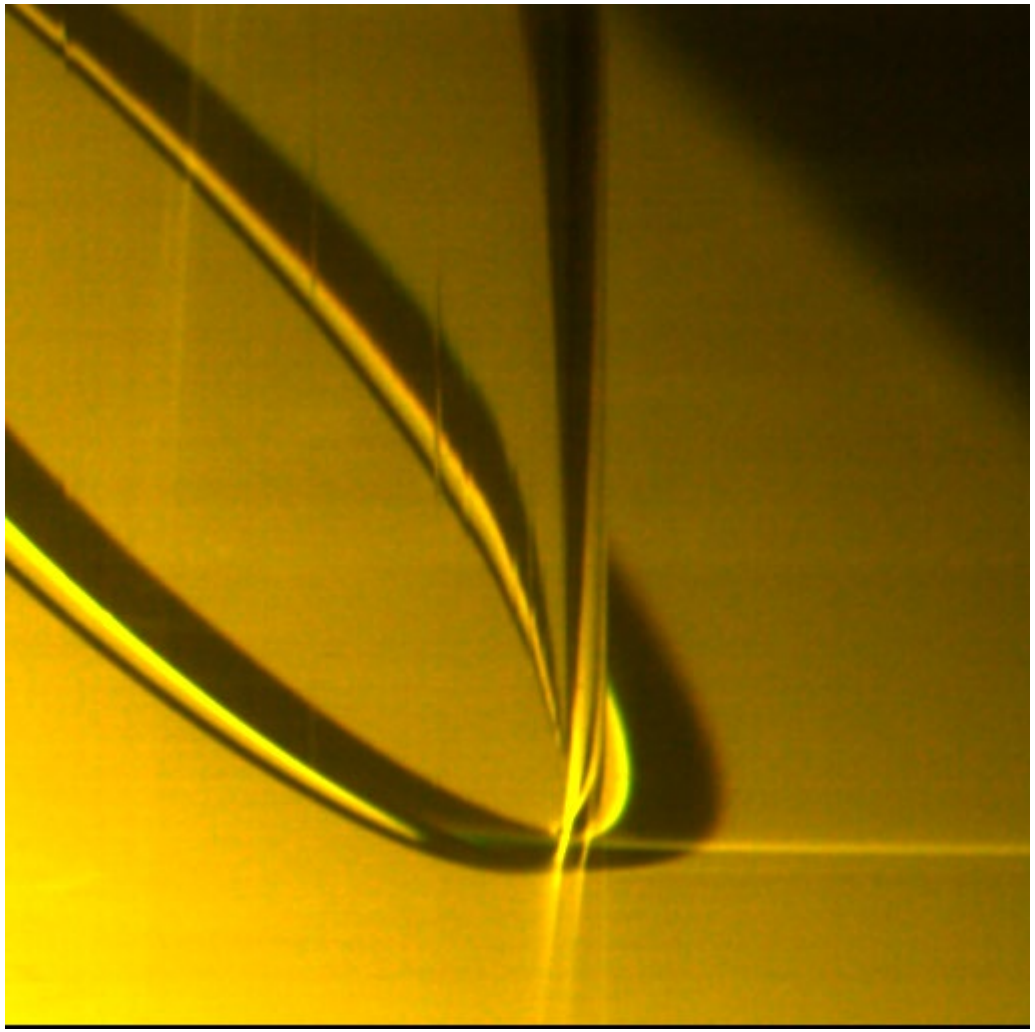
- Gaussian approximates a low-pass filter
- It smoothes the image

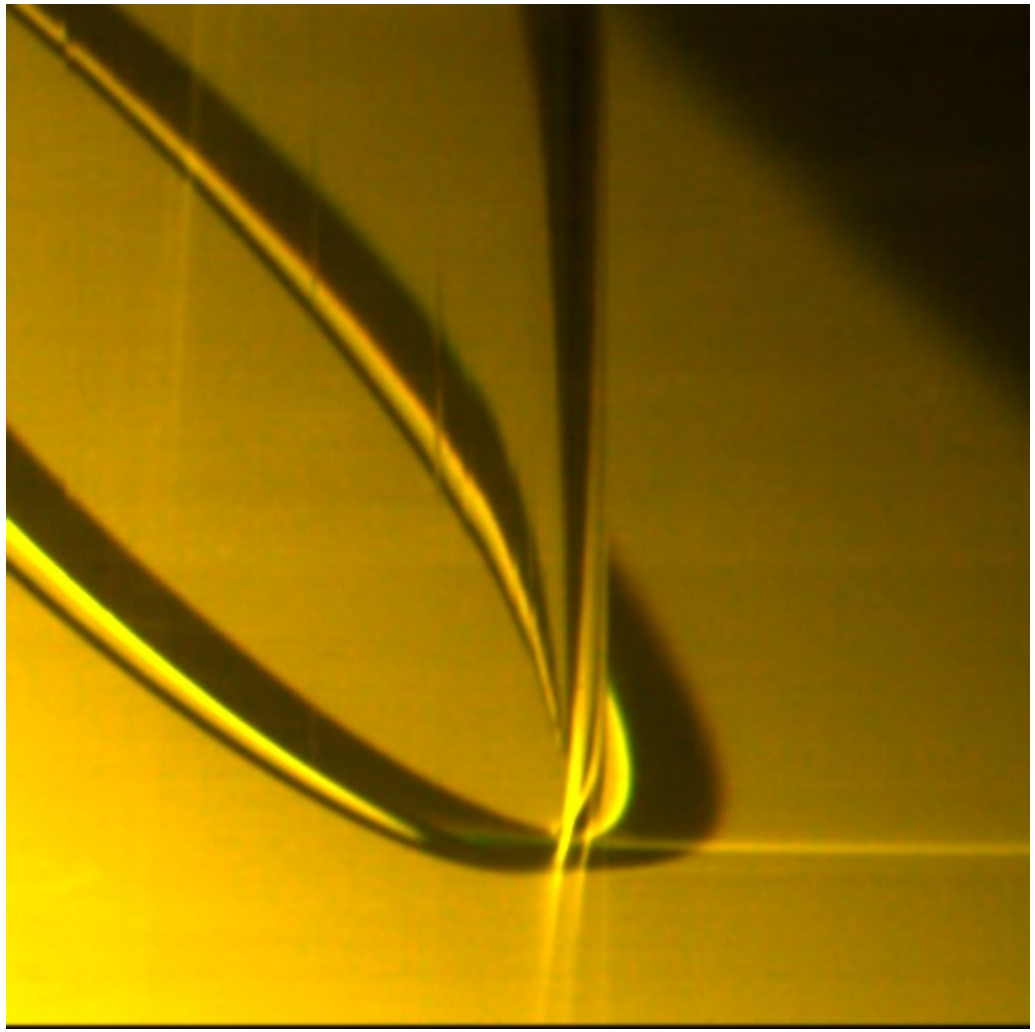
Gaussian – σ=0.2



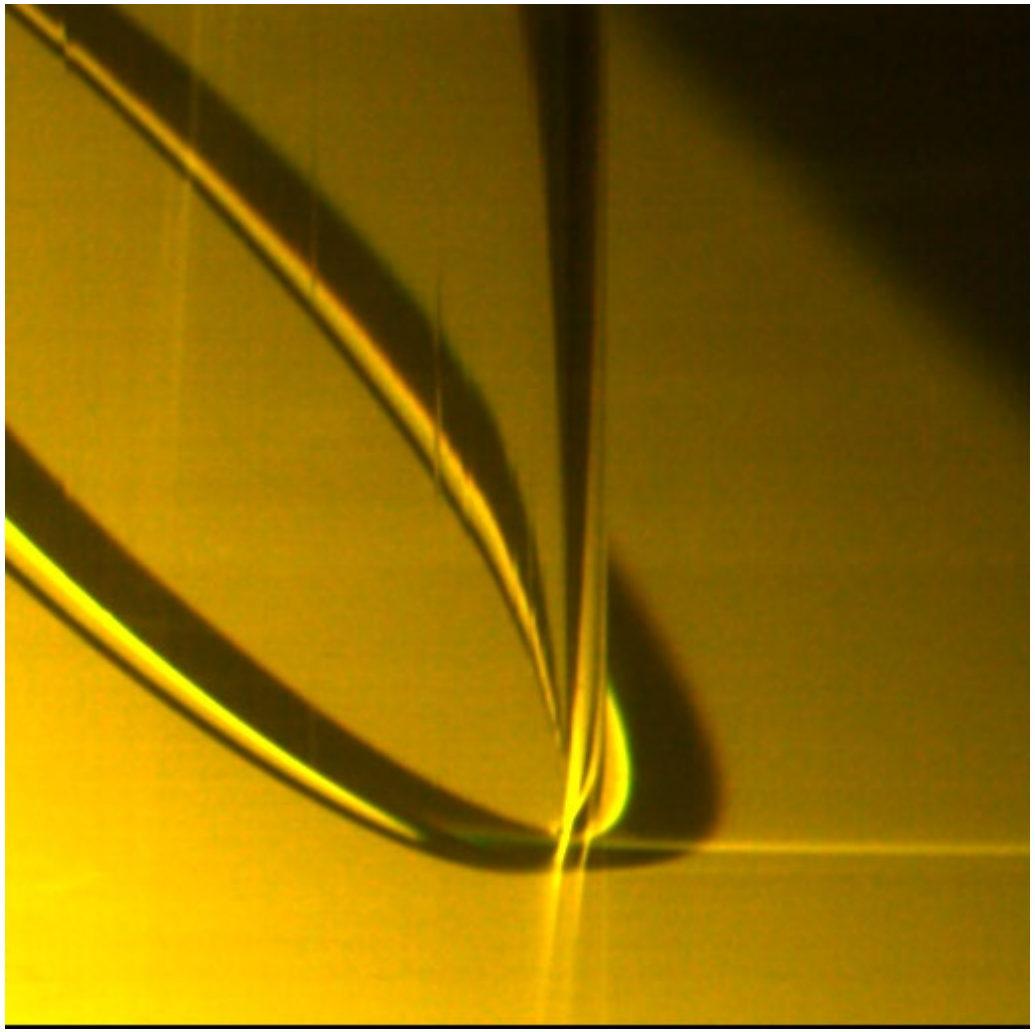


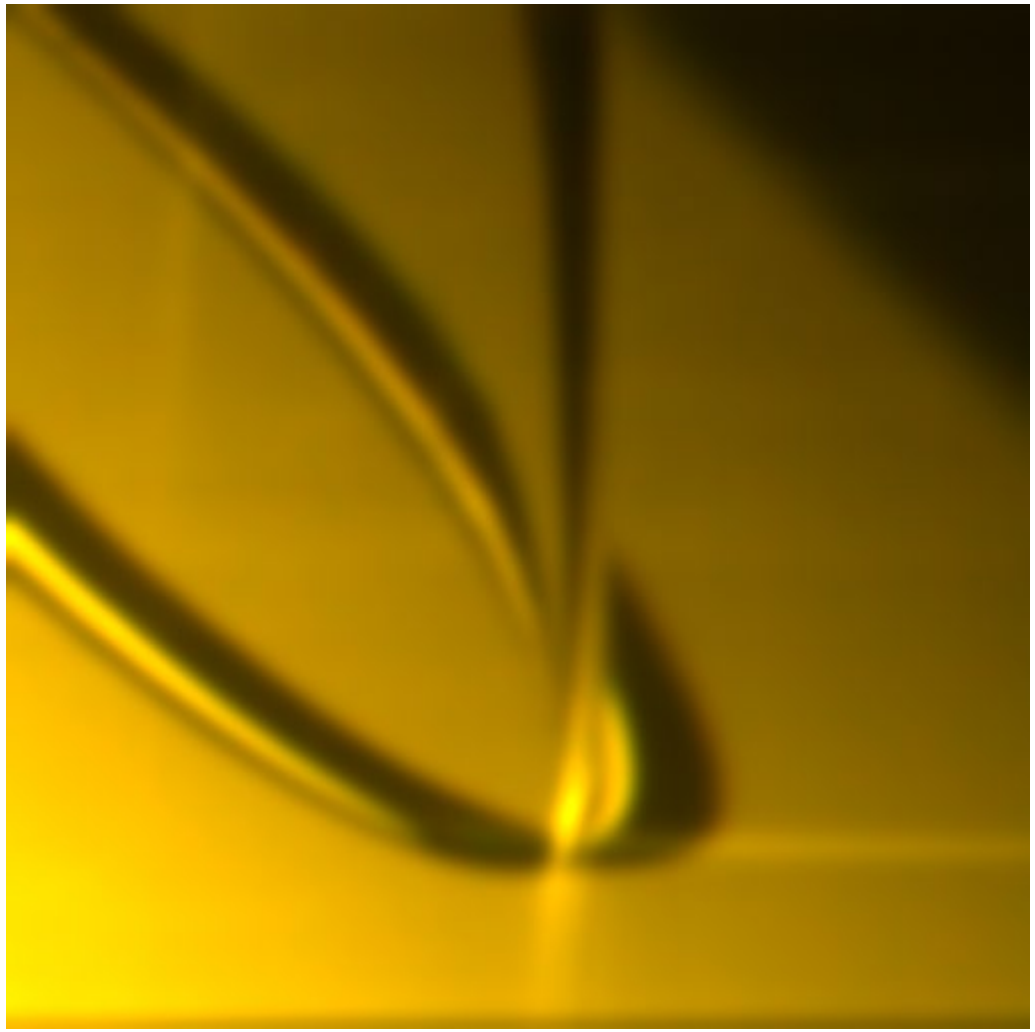
Gaussian – σ=0.8



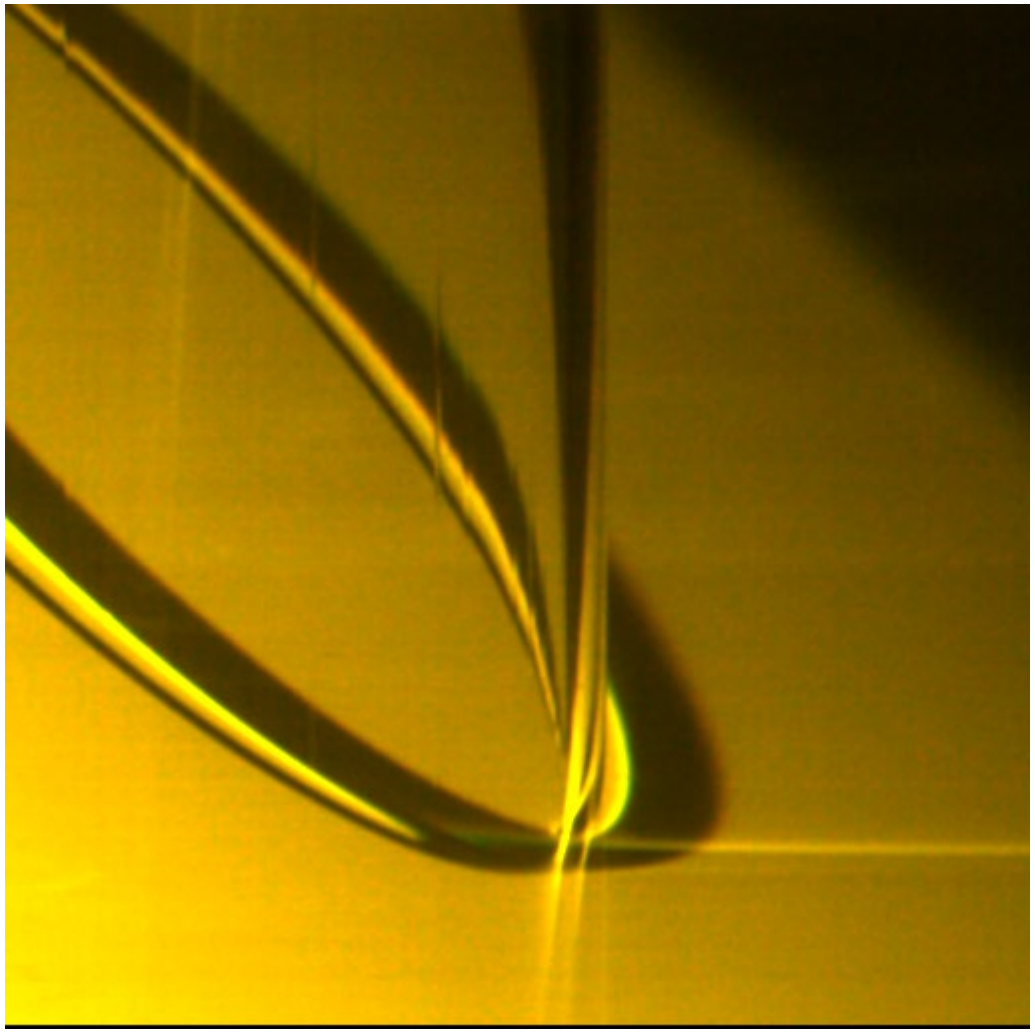


Gaussian – σ=5





Gaussian – σ=10



Gaussian - Implementation

- 2D filter mask $\sigma = 0.84$
- Width of the mask should follow 3σ rule
  - at 3σ the value of the Gaussian is less than 0.1%.

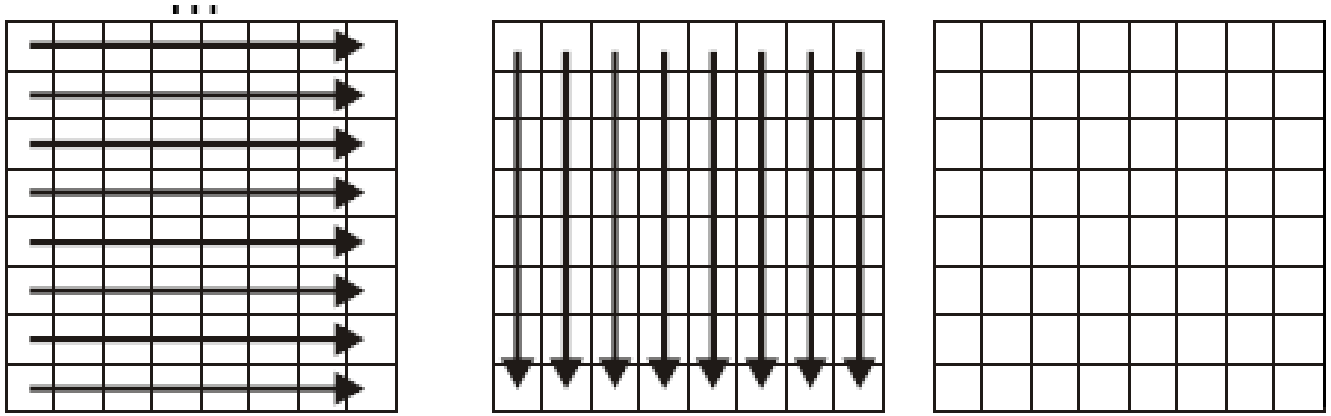
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00038771	0.01330373	0.11098164	**0.22508352**	0.11098164	0.01330373	0.00038771
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067

Gaussian - Implementation

- Gaussian filter is separable

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\bar{f}(x,y) = G_{(x,y)} \otimes f(x,y) = G_y \otimes (G_x \otimes f(x,y))$$



$G_x \otimes f(x,y)$ $G_y \otimes G_x \otimes f(x,y)$ $\bar{f}(x,y)$

Unsharp Masking

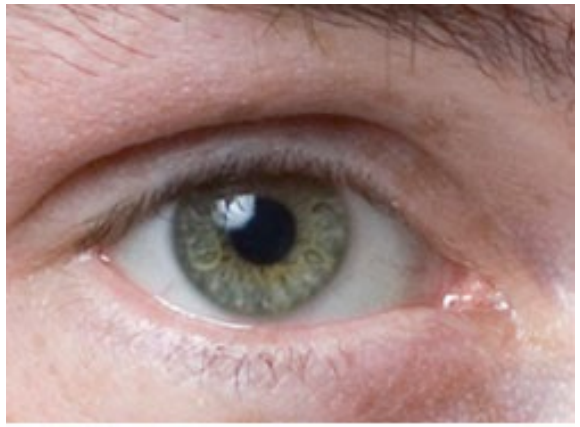
- Sharpen the image by subtracting the low frequency (unsharpened or blurred) component

$$U(x,y) = G_{(x,y)} \otimes f(x,y)$$

$$\bar{f}(x,y) = \frac{c}{2c-1} f(x,y) - \frac{(1-c)}{2c-1} U(x,y)$$

Unsharp Masking

[Wikipedia]

s1  - s2  = 

sharpened

Sobel Operator

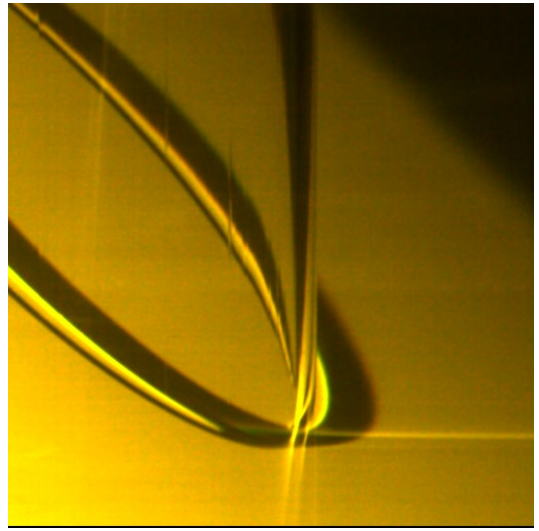
- A simple edge detection filter for horizontal and vertical edges

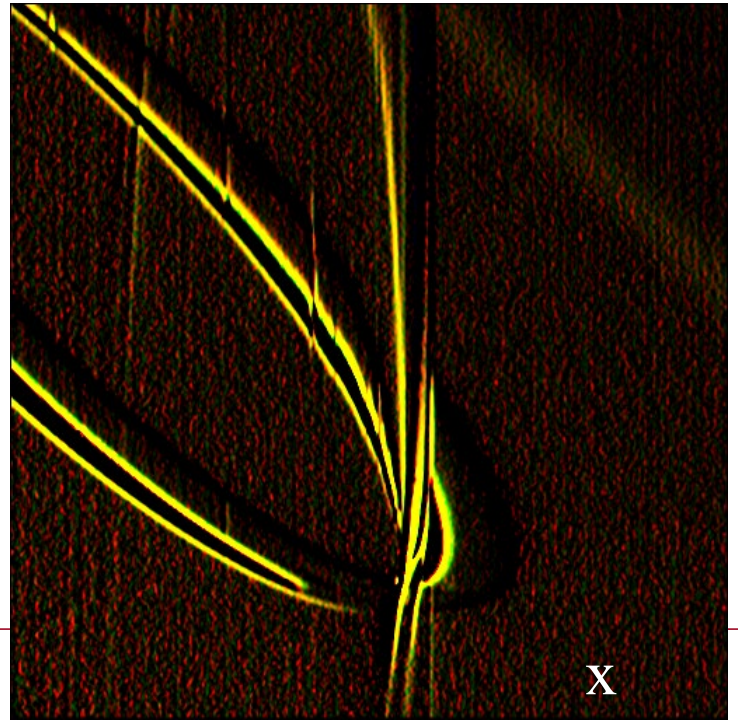
$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Arbitrary direction (high-pass filter) approximated by

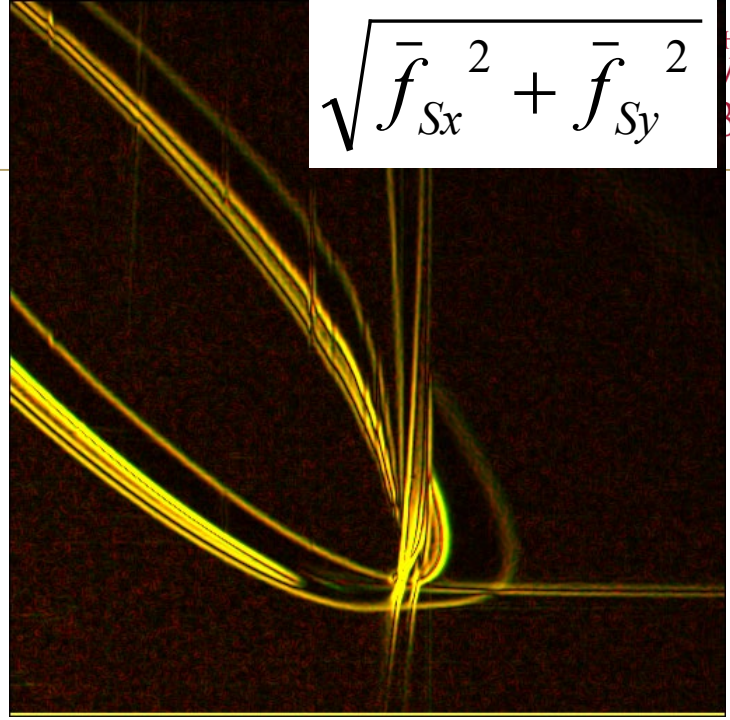
$$\sqrt{\bar{f}_{Sx}^{\,2} + \bar{f}_{Sy}^{\,2}}$$

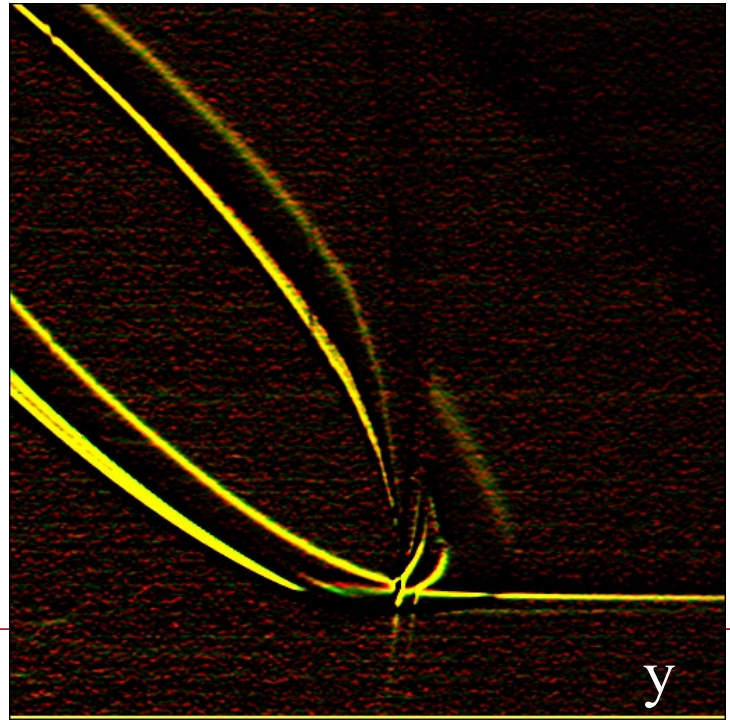
Sobel Operator





$$\sqrt{\bar{f}_{Sx}^{\,2} + \bar{f}_{Sy}^{\,2}}$$



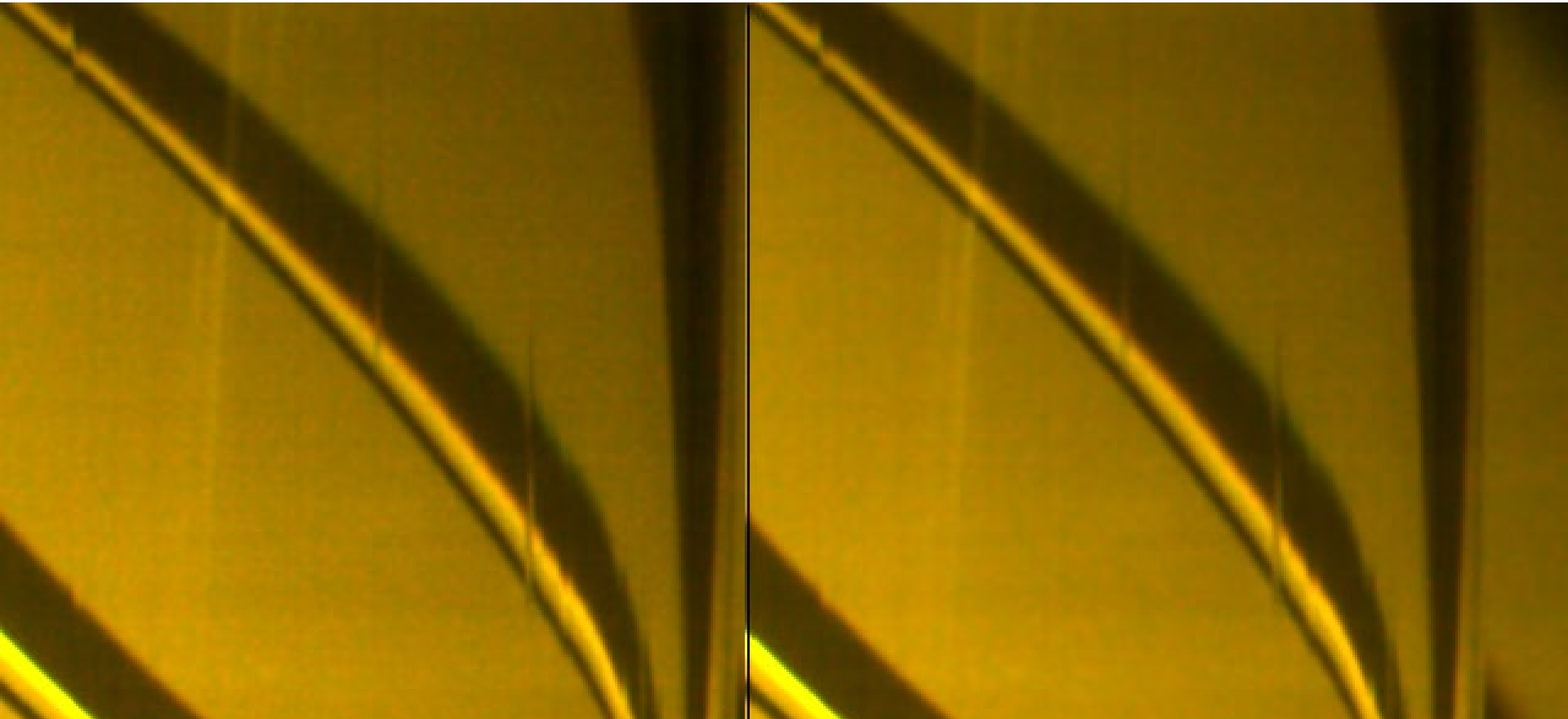


Non-linear Filters: Median

- Take all samples in under the filter mask, sort them
- Store back the median, element at position (window*window/2)
- Nicely removes outliers and “sticky pixels”

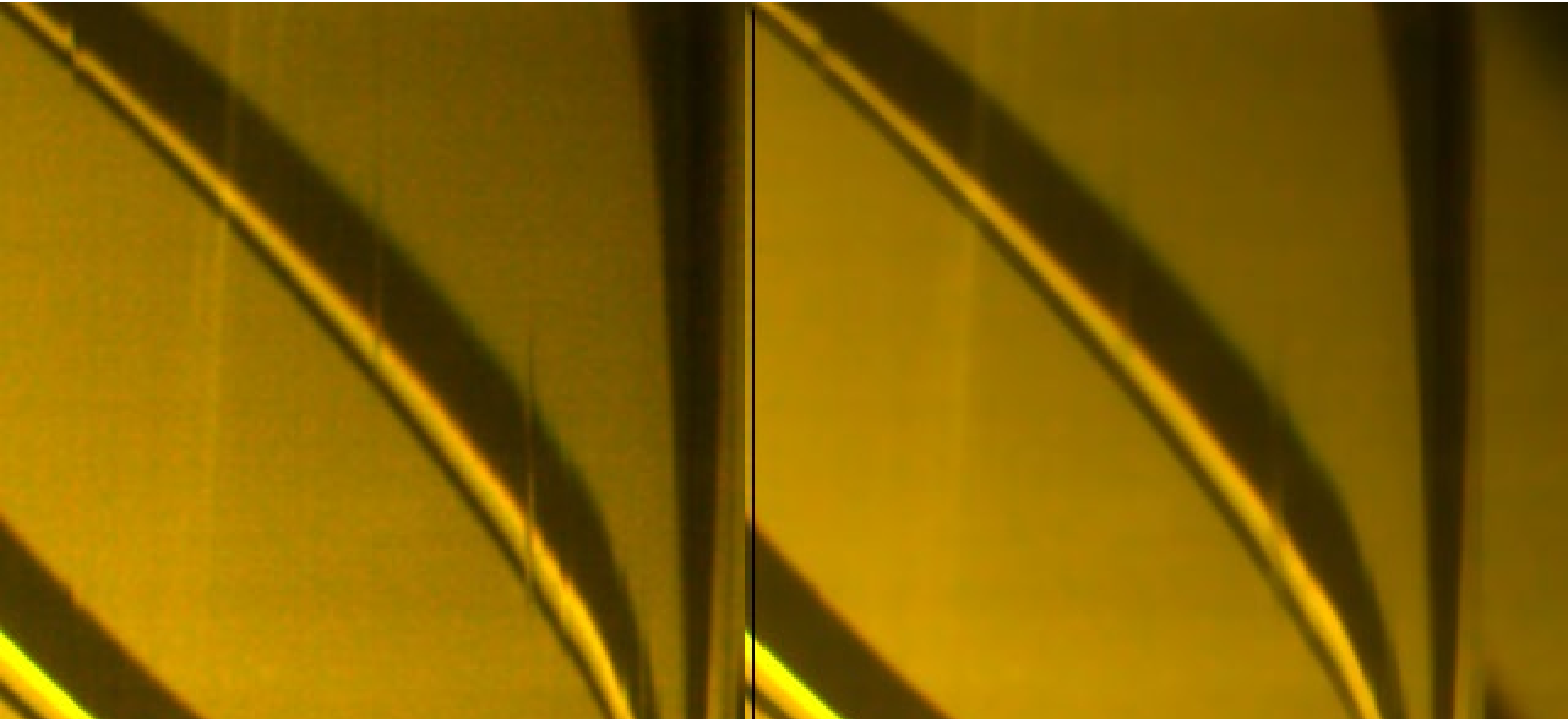
- Non-separable but lots and lots of accelerations

Median 3x3



Median 7x7

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN



Bilateral Filter

Bilateral Filtering

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Gaussian filter kernel - spatial distance

$$g(p_1, p_2) = e^{-\frac{1}{2}\left(\frac{d(p_1 - p_2)}{\sigma_g}\right)^2}$$

- Bilateral filter kernel – spatial and value distance

$$b(p_1, p_2) = g(p_1, p_2) \cdot e^{-\frac{1}{2}\left(\frac{d(f(p_1) - f(p_2))}{\sigma_b}\right)^2}$$

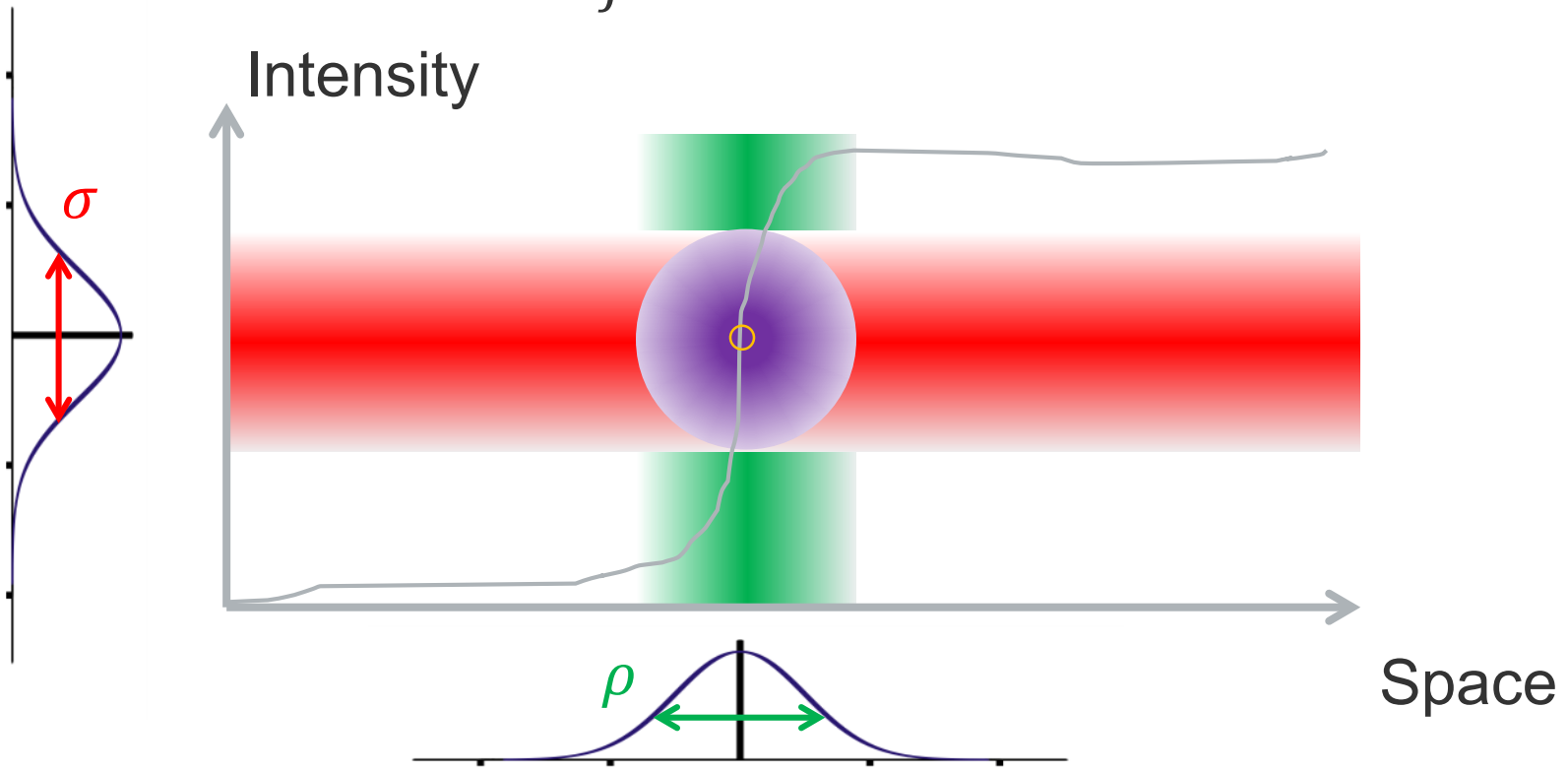
  - prevents blurring across edges

  - for two points p_1, p_2 under the filter mask and

  - some distance function d

Bilateral Filtering

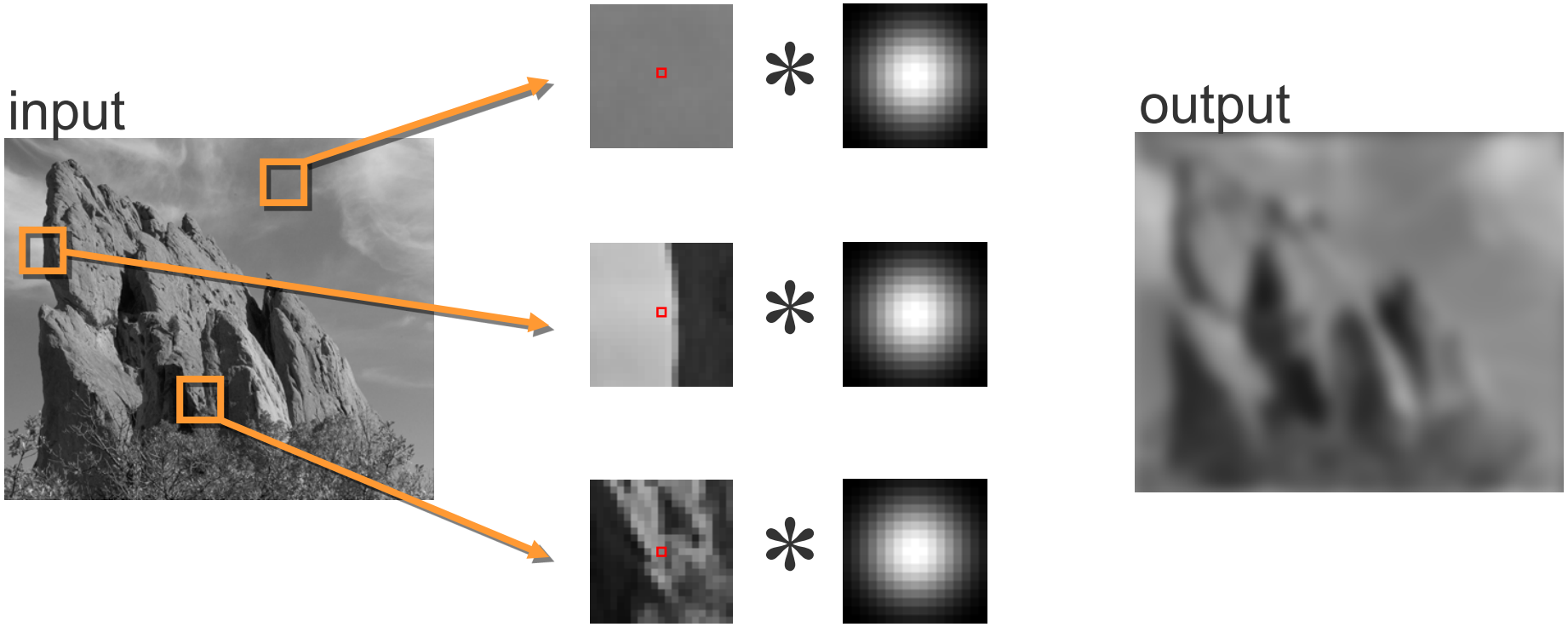
$$\hat{x}(i) = \frac{1}{C_i} \sum_j y(j) e^{-\frac{\|i-j\|^2}{2\rho^2}}$$



Smith and Brady (1997)

Gaussian Smoothing

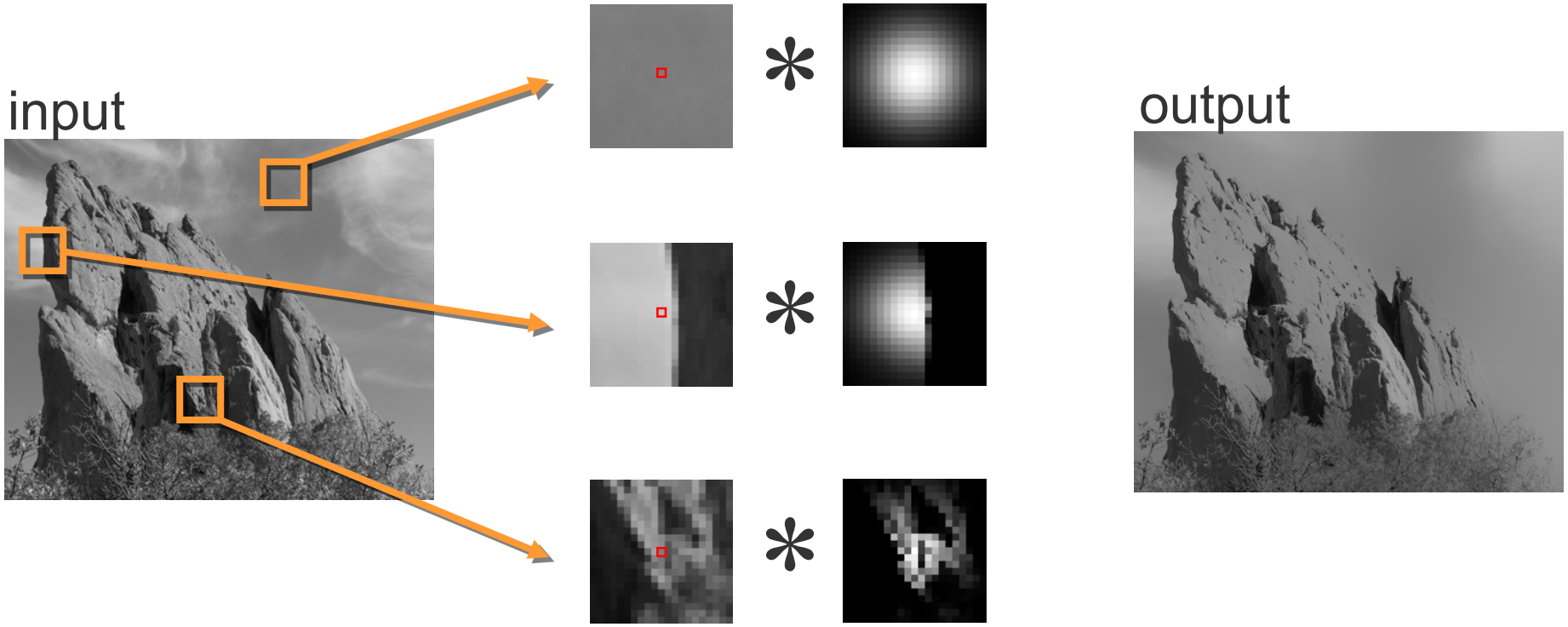
Slides taken from Sylvain Paris, Siggraph 2007



Same Gaussian kernel everywhere
Averages across edges ⇒ blur

Bilateral Filtering

Slides taken from Sylvain Paris, Siggraph 2007

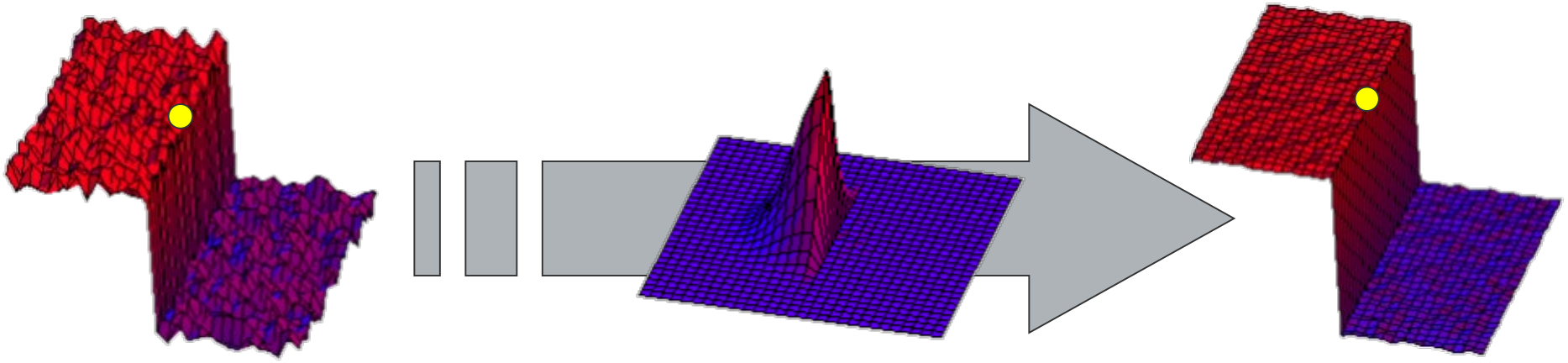


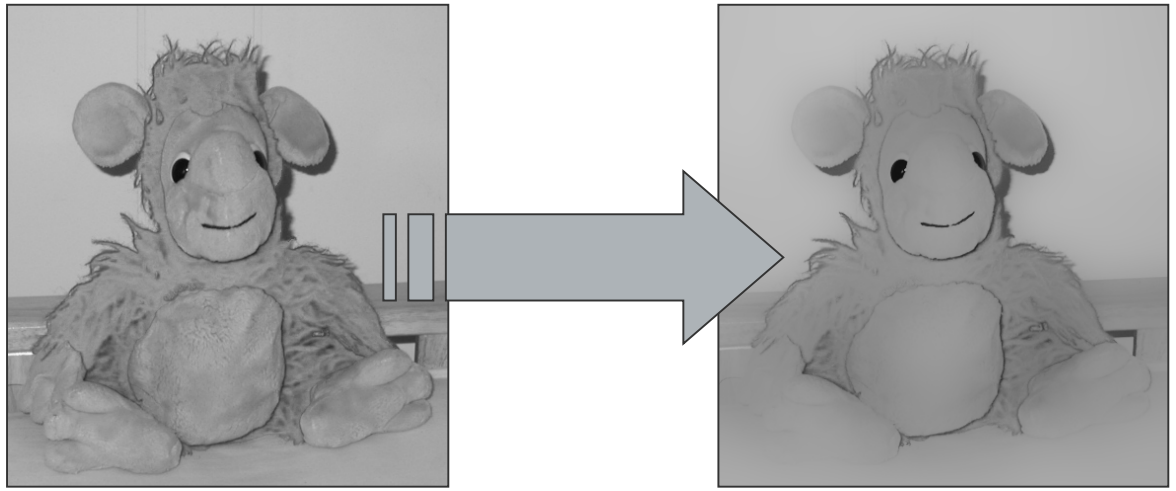
Kernel shape depends on image content
Avoids averaging across edges

Large-scale Layer

- Bilateral filter – edge preserving filter

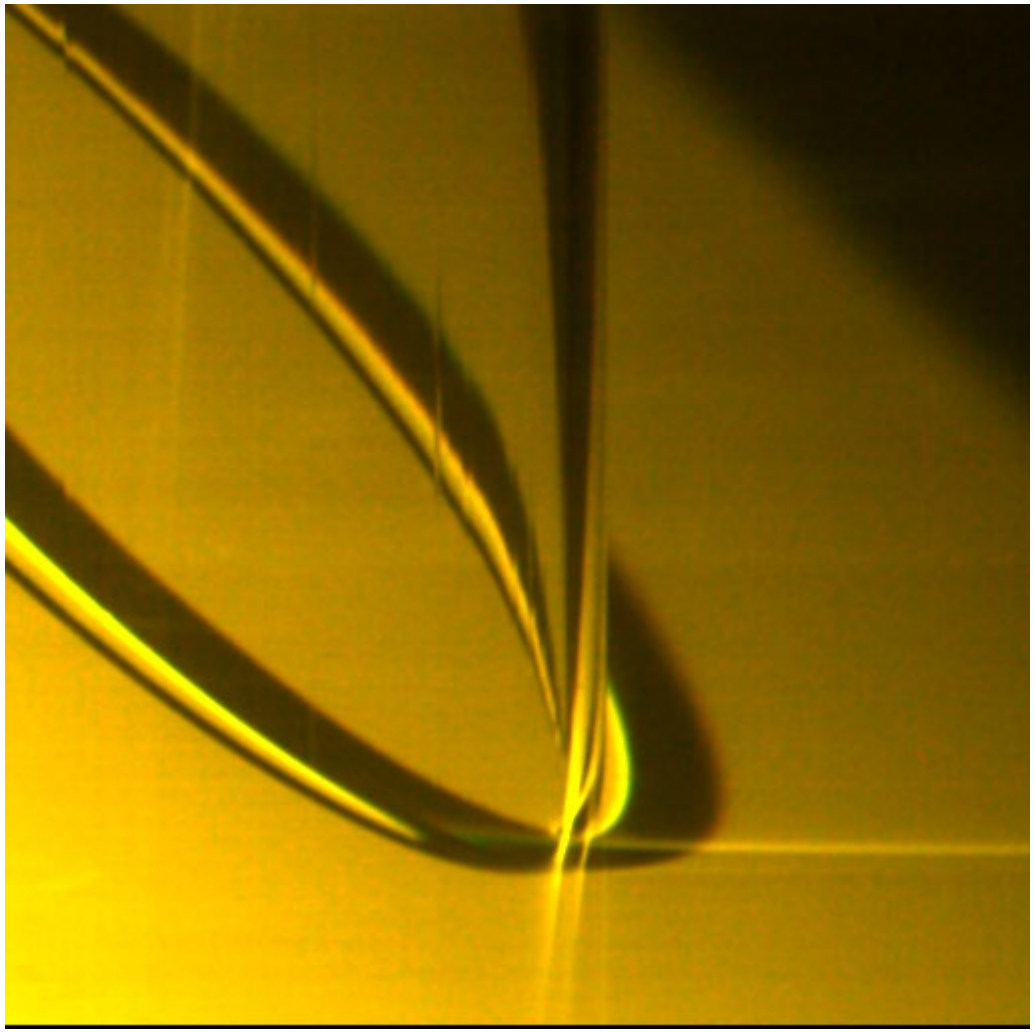
Smith and Brady 1997; Tomasi and Manducci 1998; Durand et al. 2002

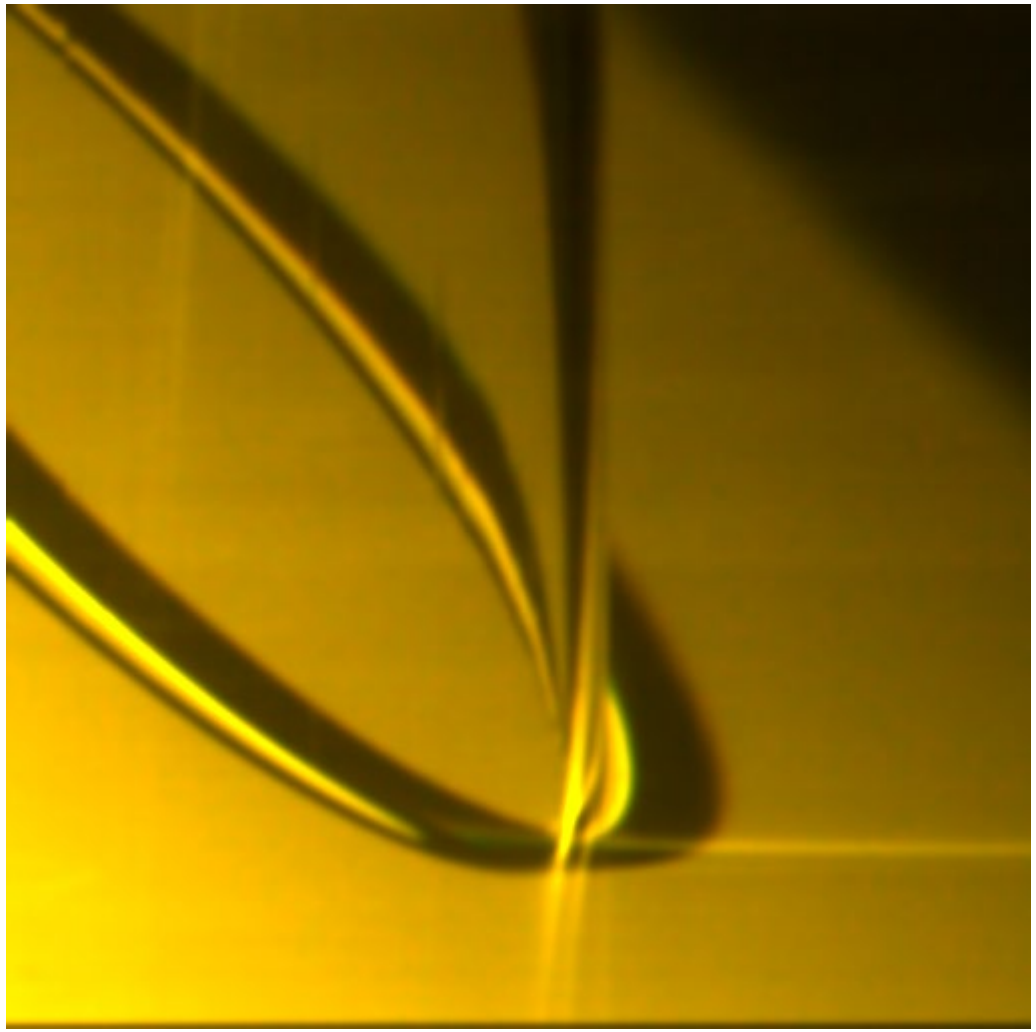




Bilateral Filter – σ_v=0.1

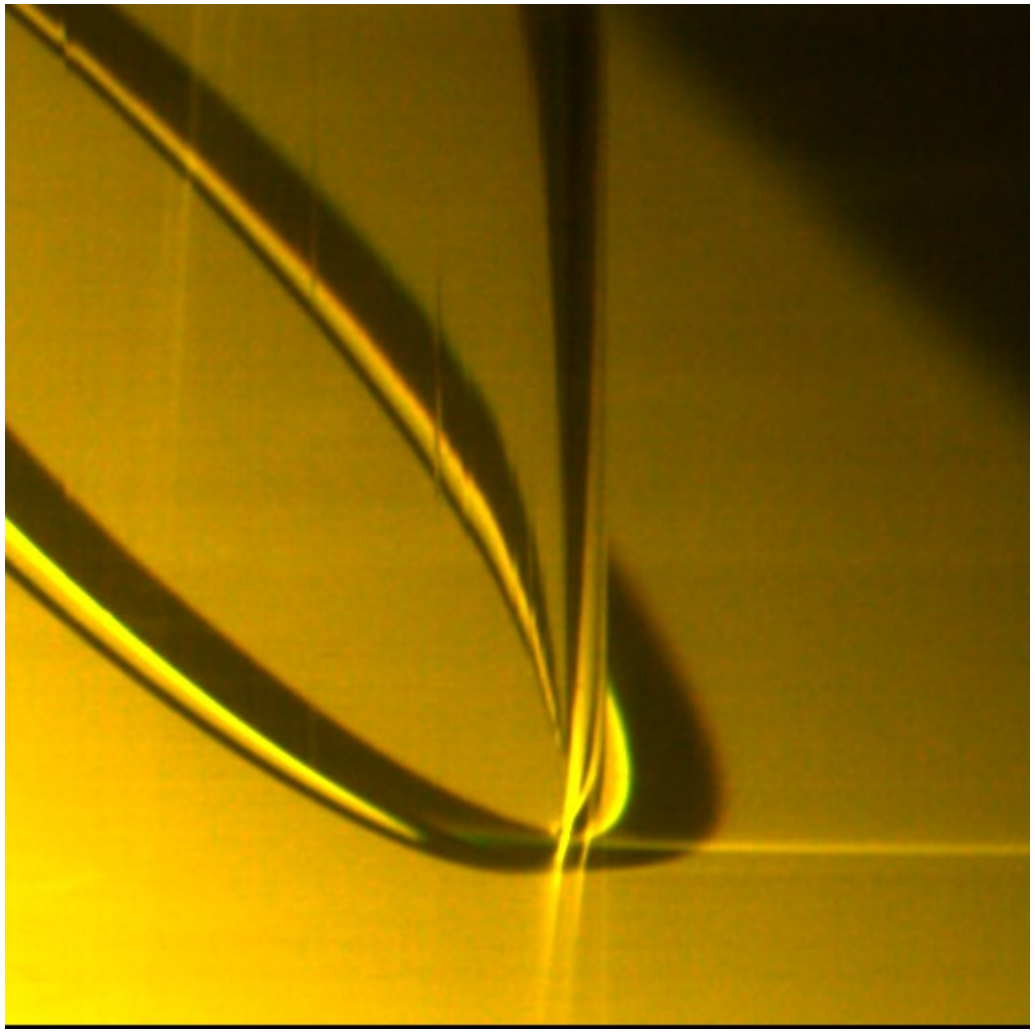
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

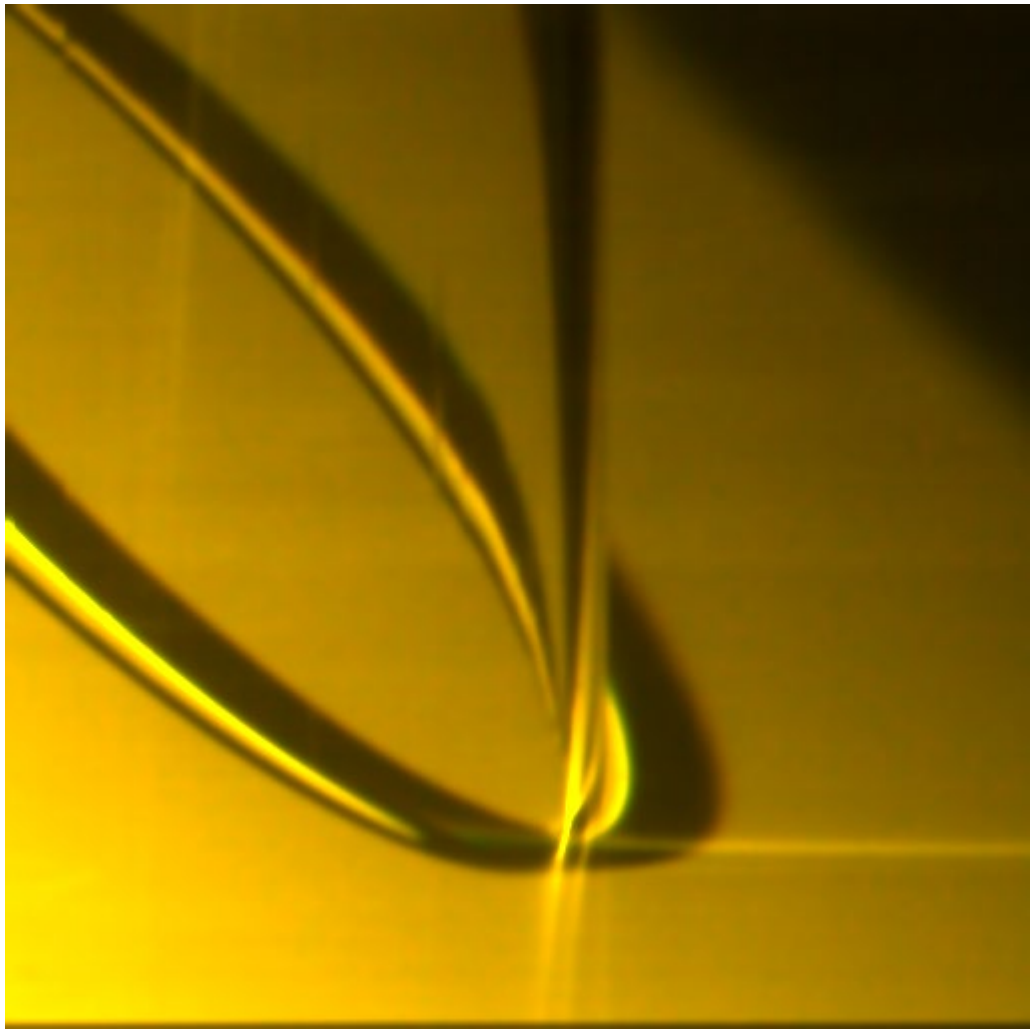




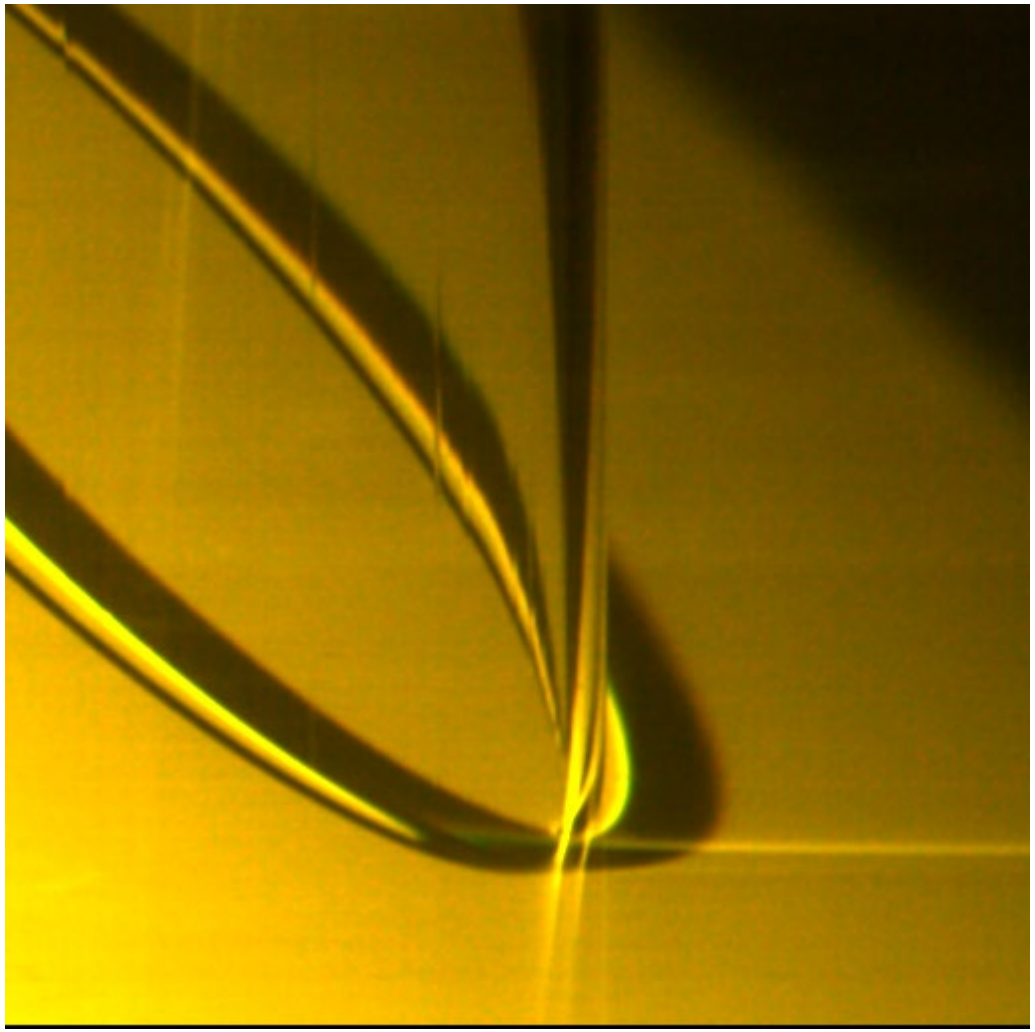
Bilateral Filter – σ_v=0.01

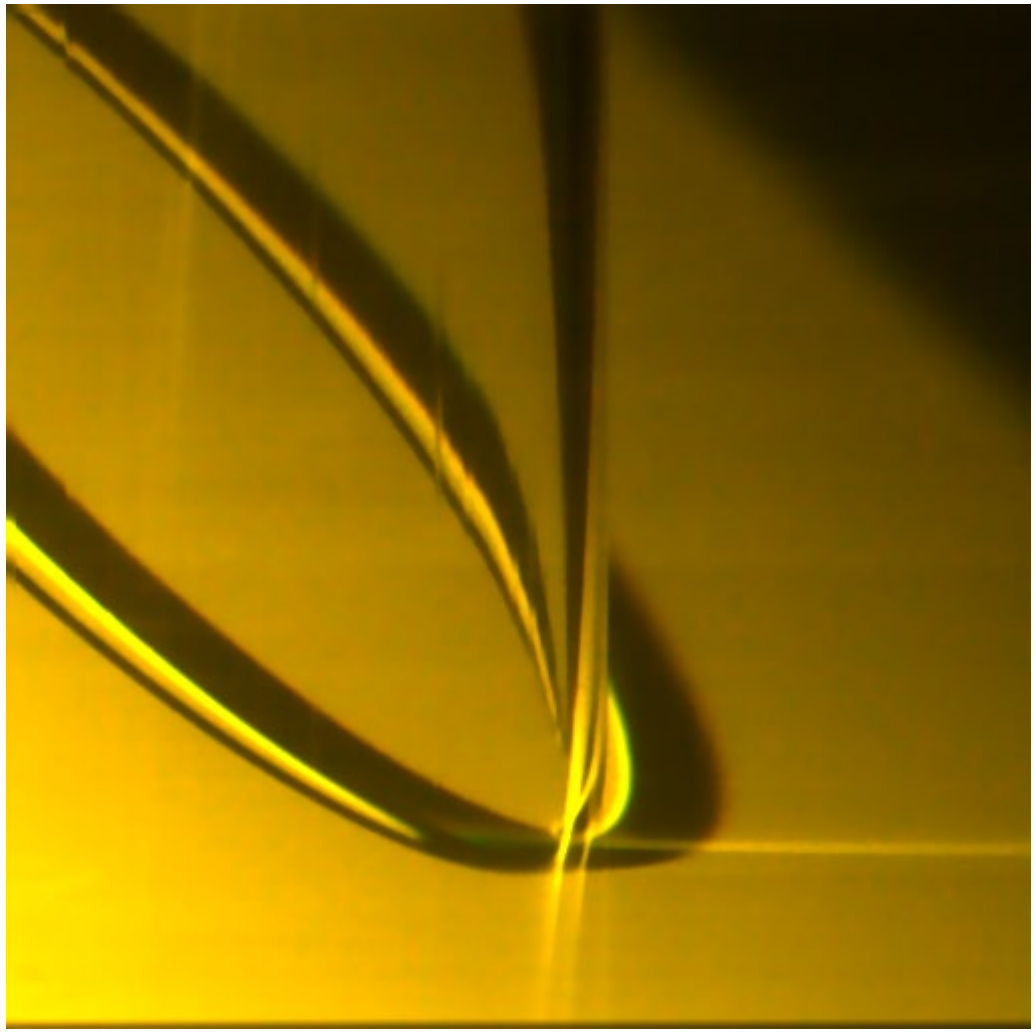
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN





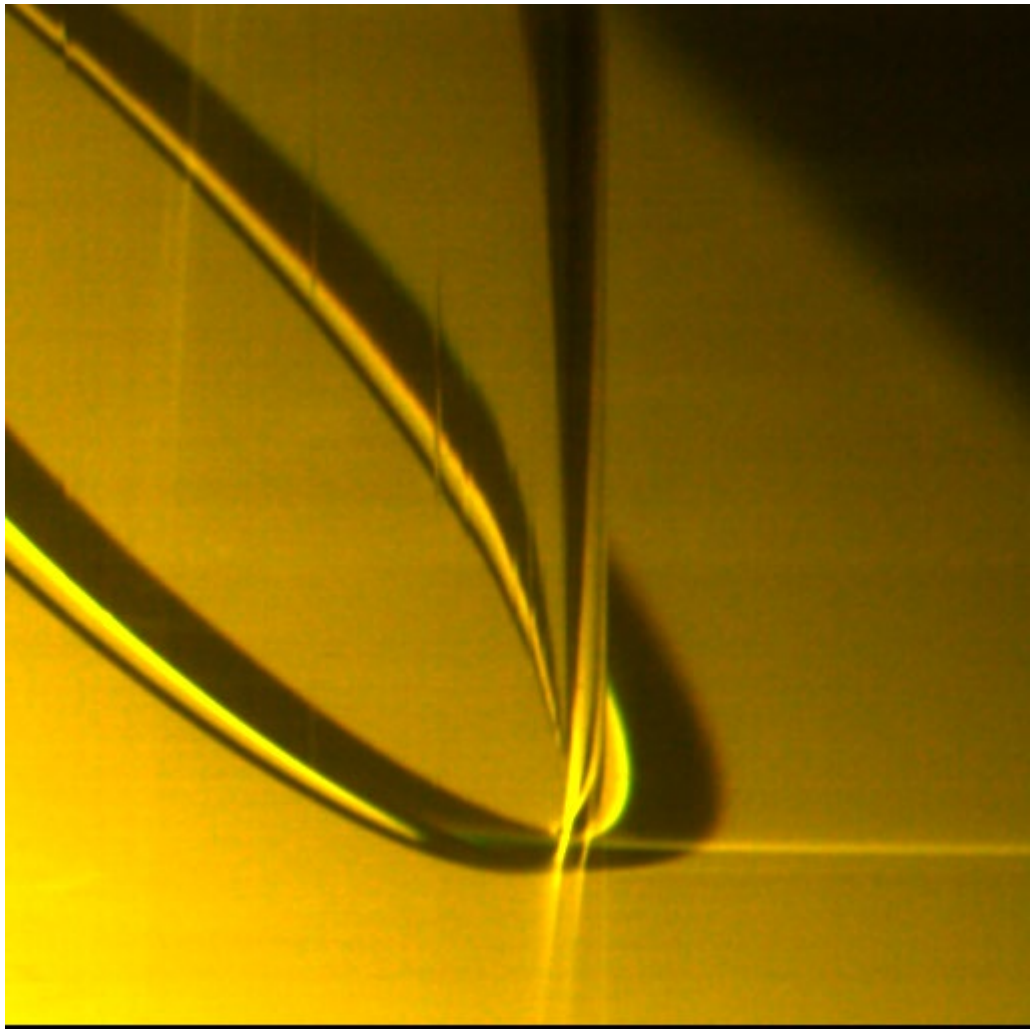
Bilateral Filter – σ_v=0.001

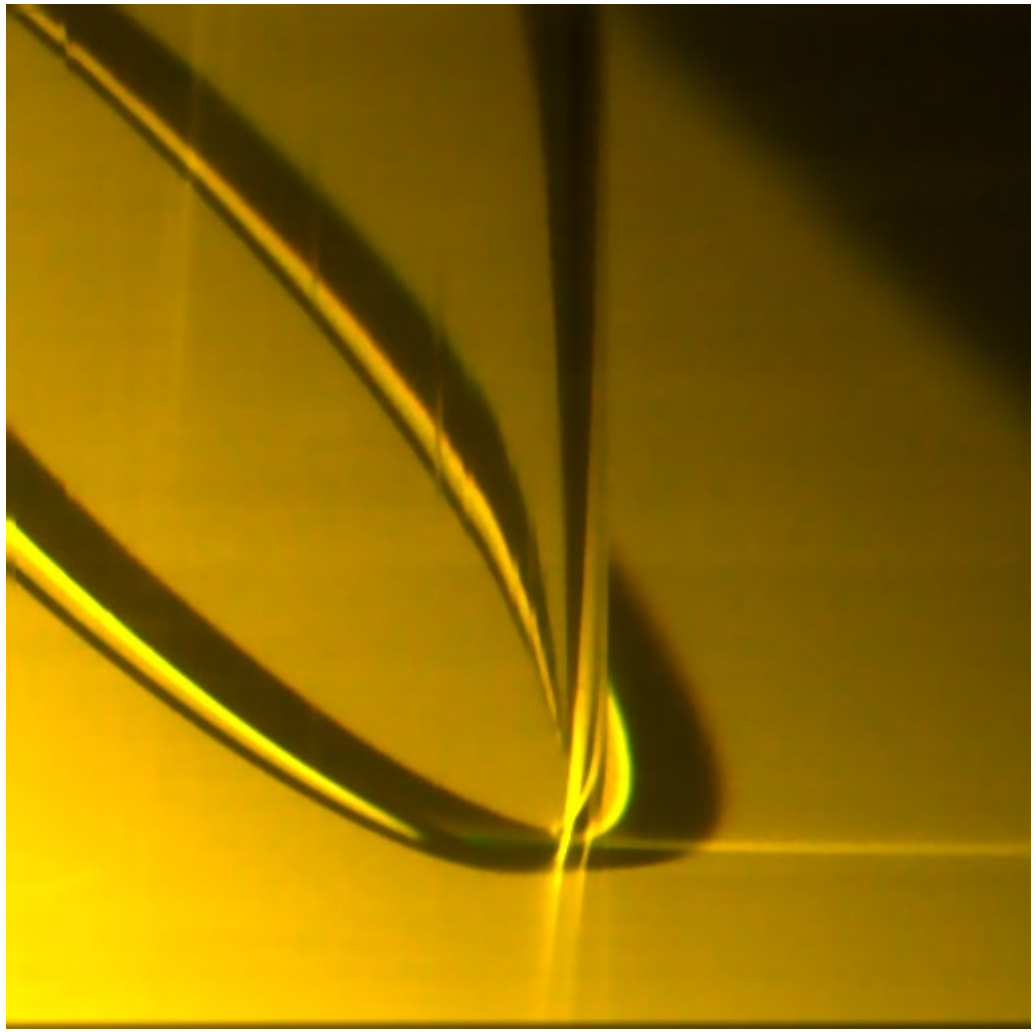




Bilateral Filter – σ_v=0.001

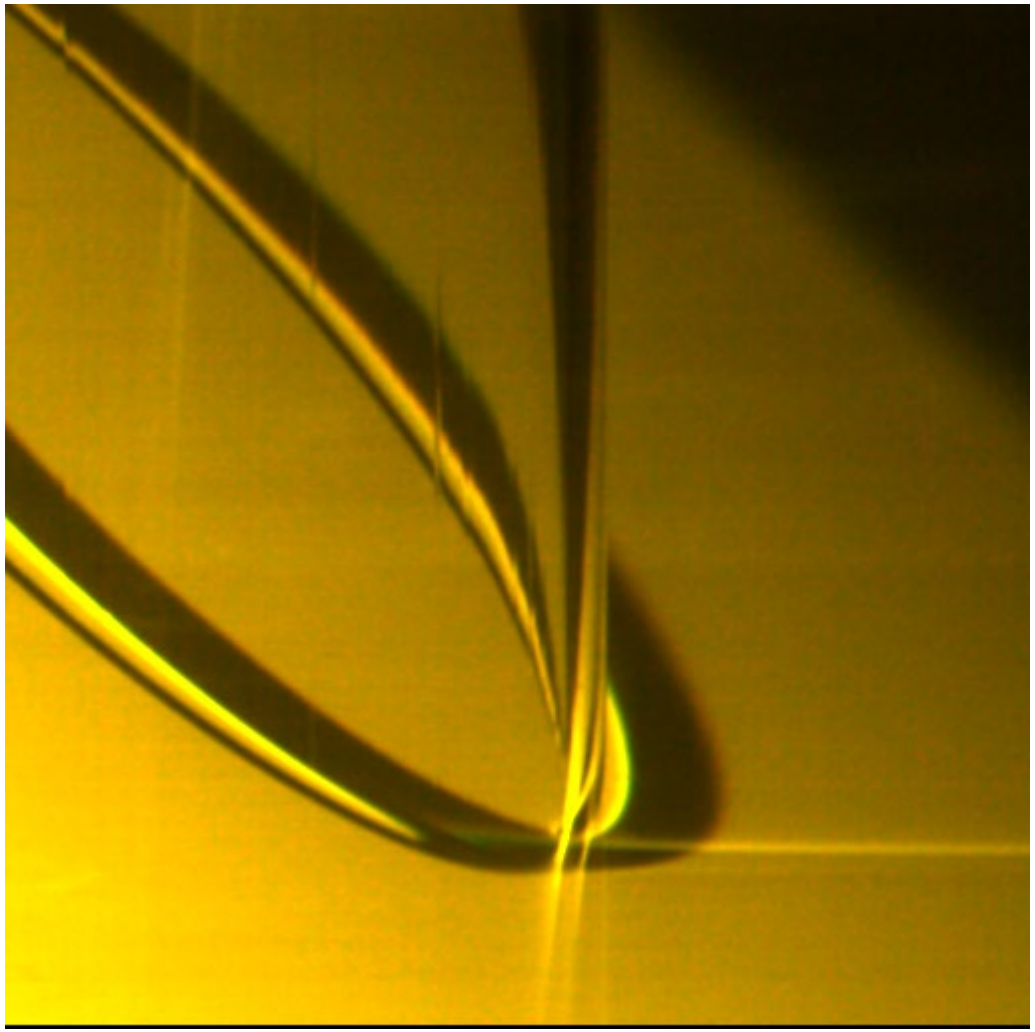
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

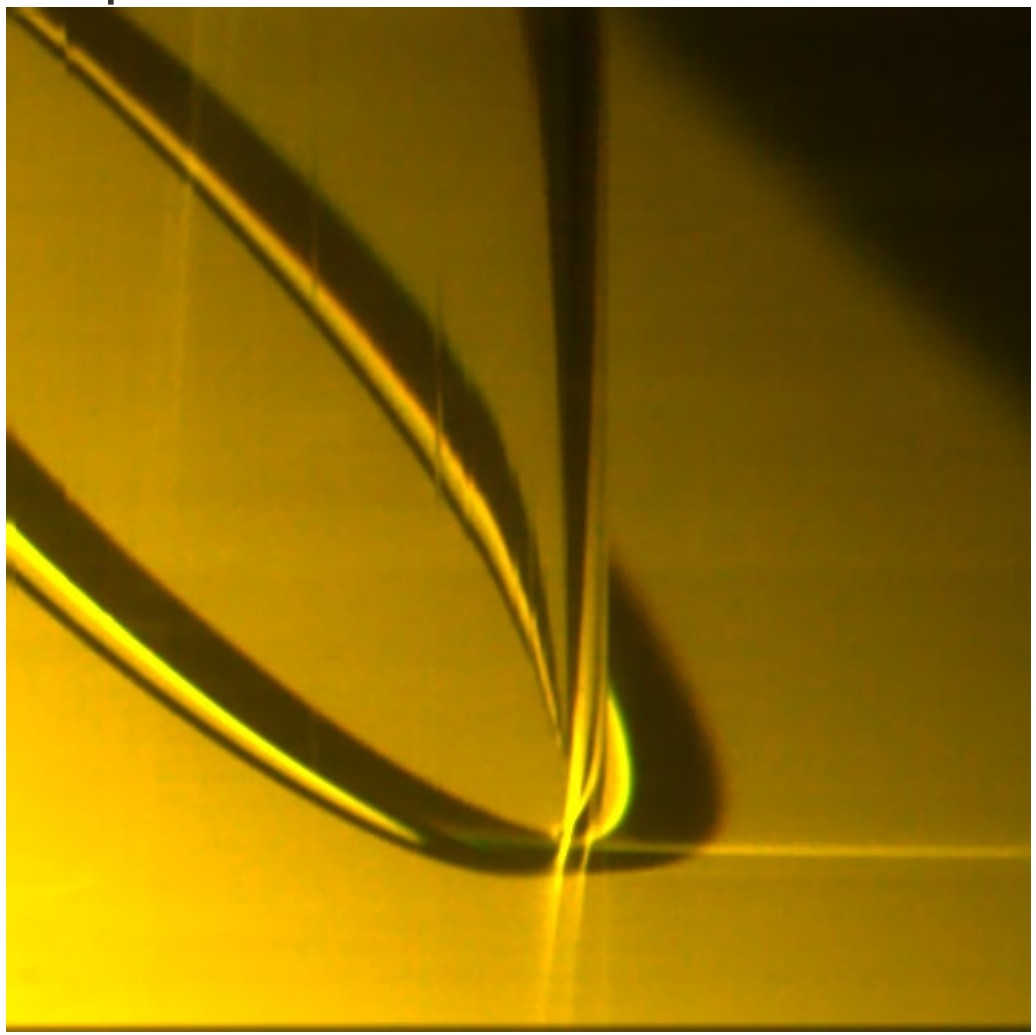




Bilateral Filter – σ_v=0.0001

Removes the noise but keeps the structure.





Flash-No-Flash [Eisemann&Durand04]

- [the following slides are from Eisemann's SIGGRAPH talk]

Available light:
+ nice lighting

- noise/blurriness
- color

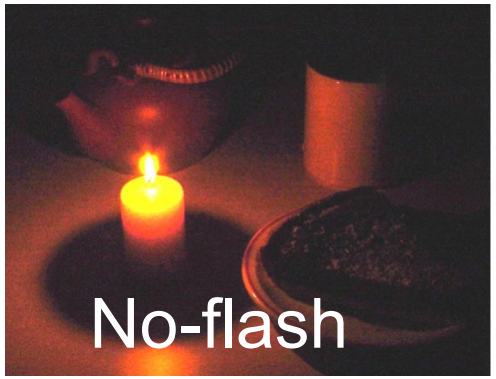
Flash

Flash:
+ details
+ color

- flat/artificial
- flash shadows
- red eyes

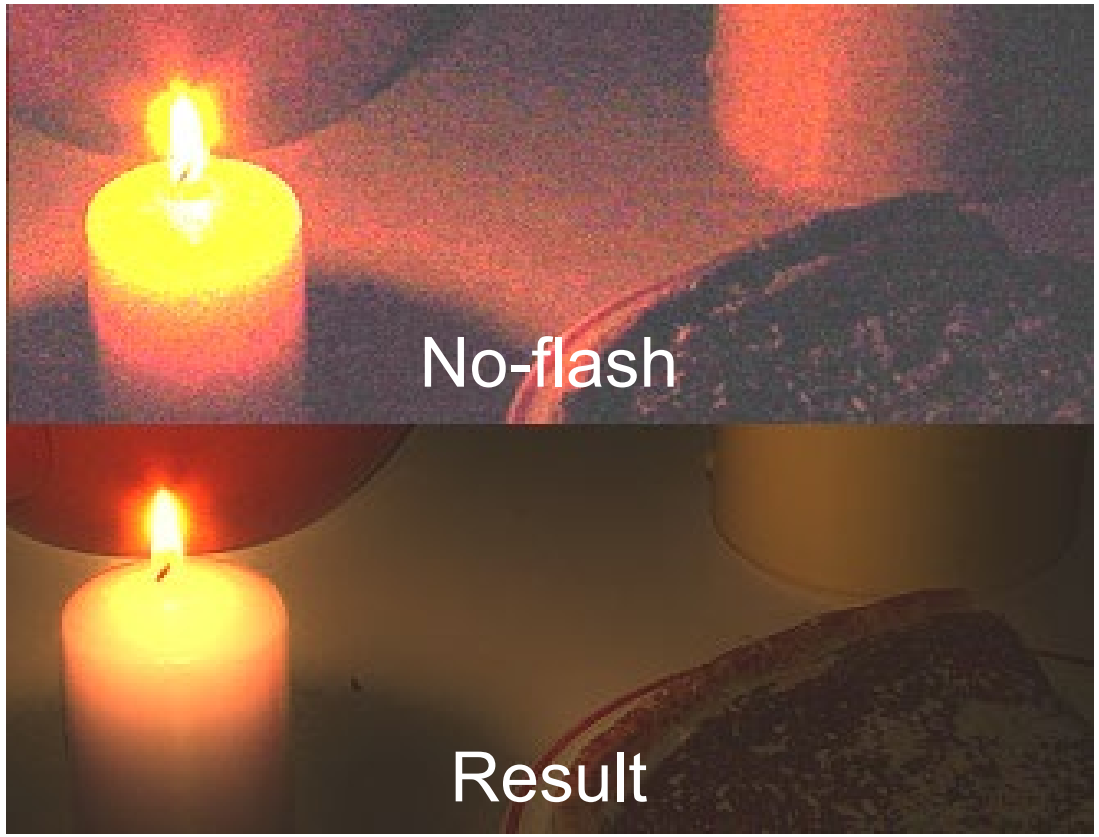
Combined

- Use no-flash image relight flash image



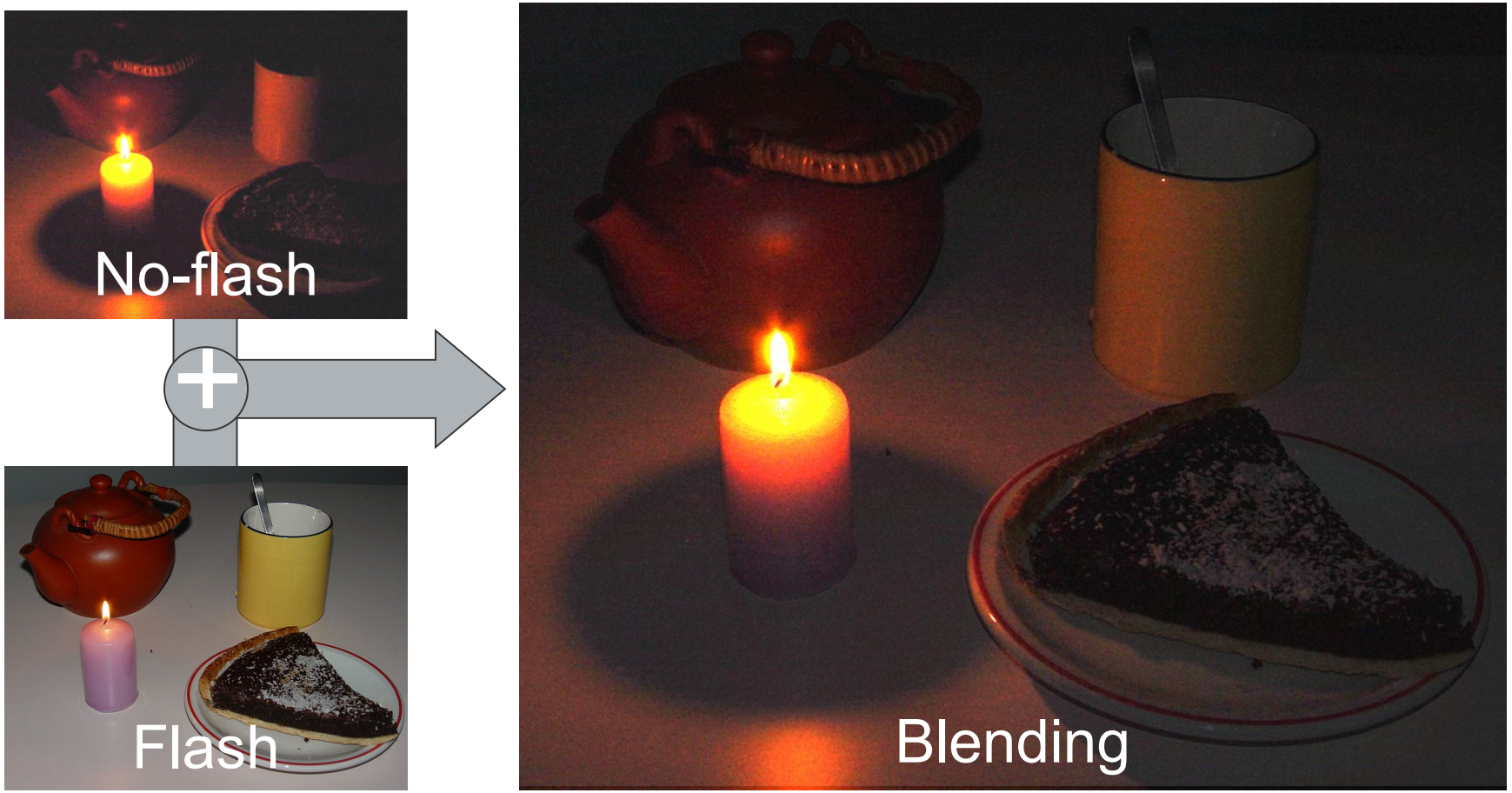
Introduction

- Use no-flash image relight flash image



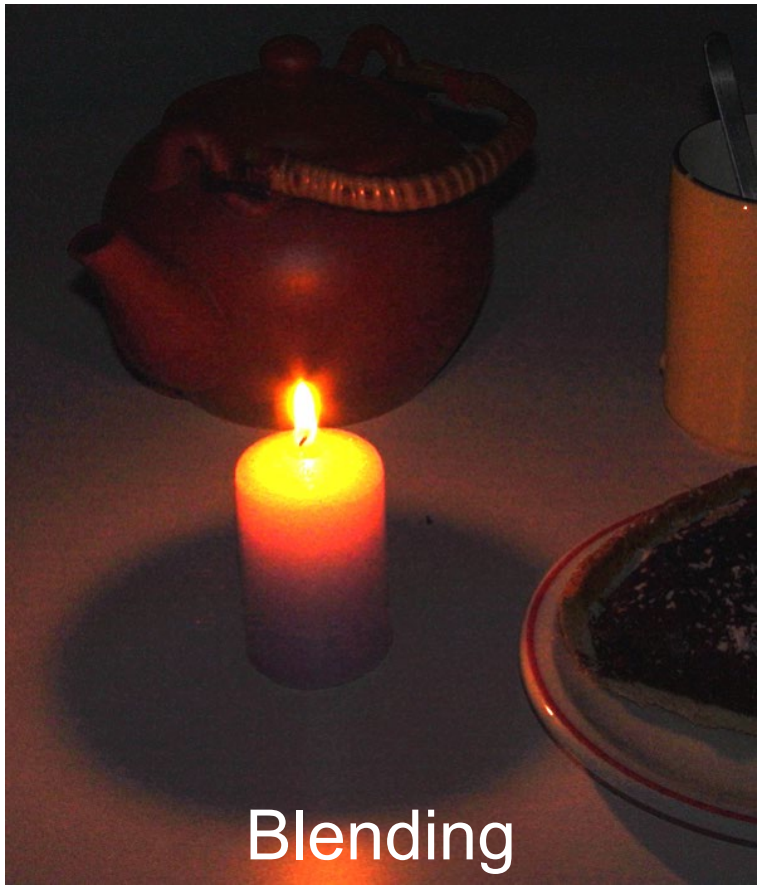
Introduction

- One approach: Blend the two photos



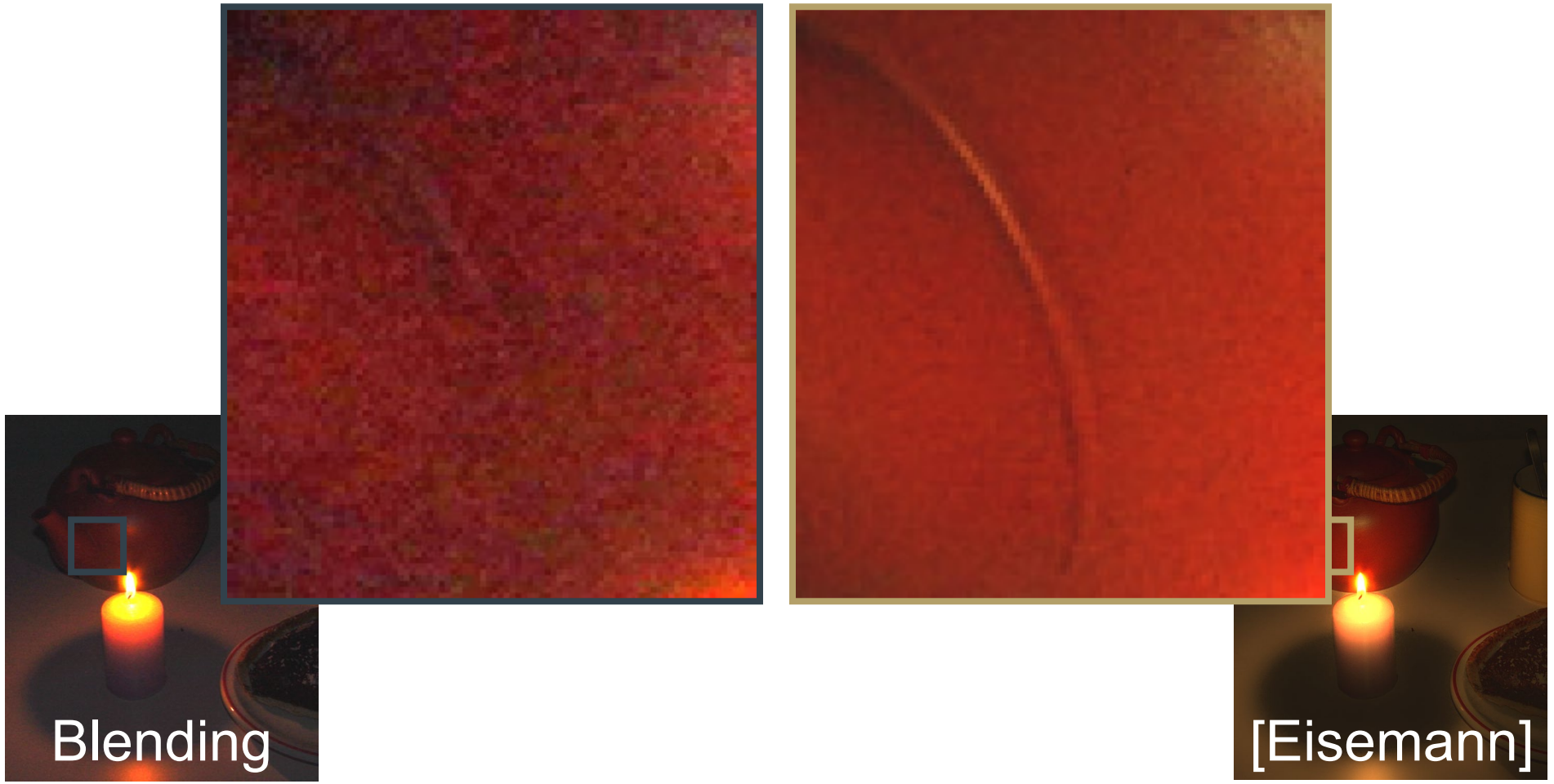
Introduction

- One approach: Blend the two photos

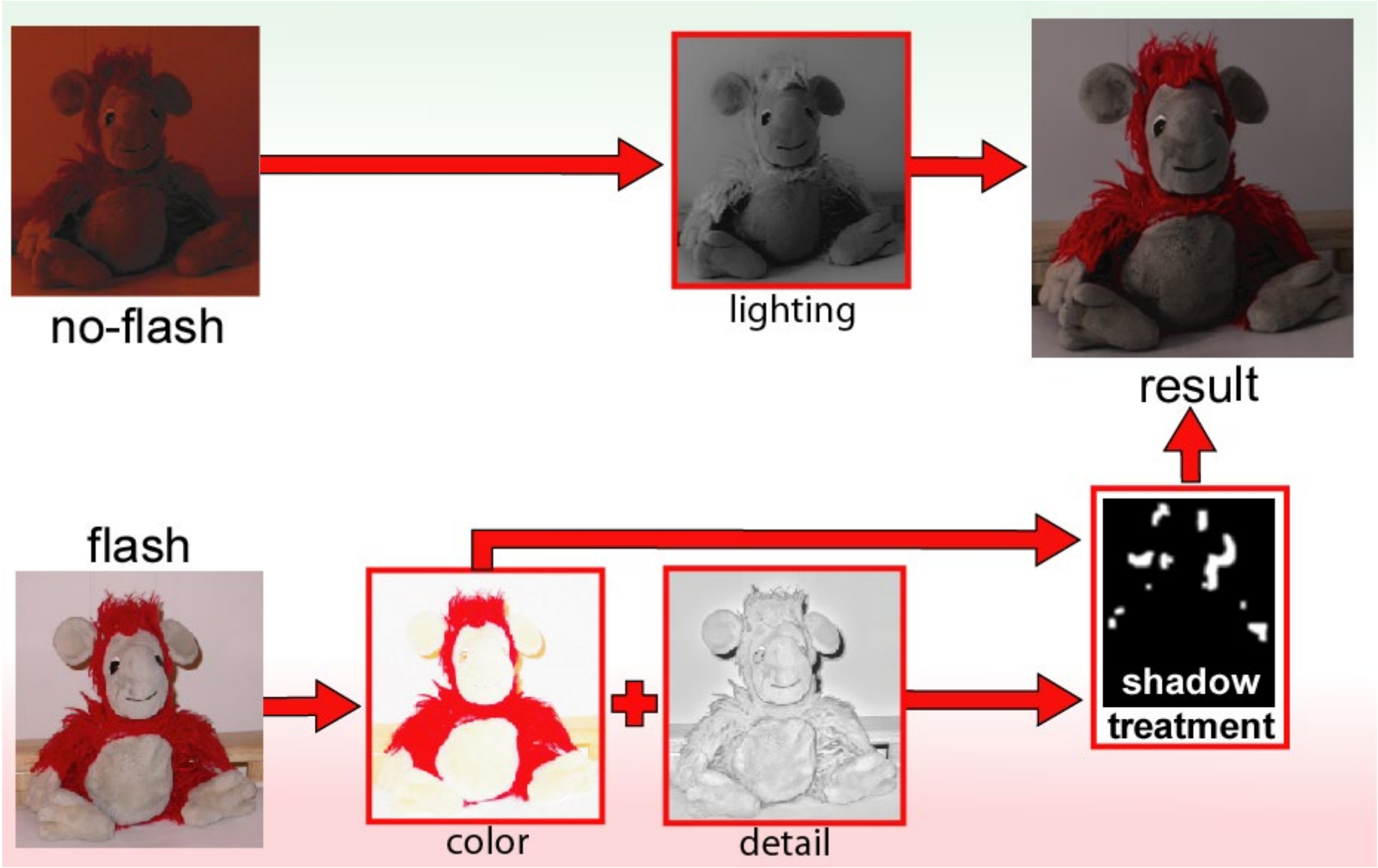


Introduction

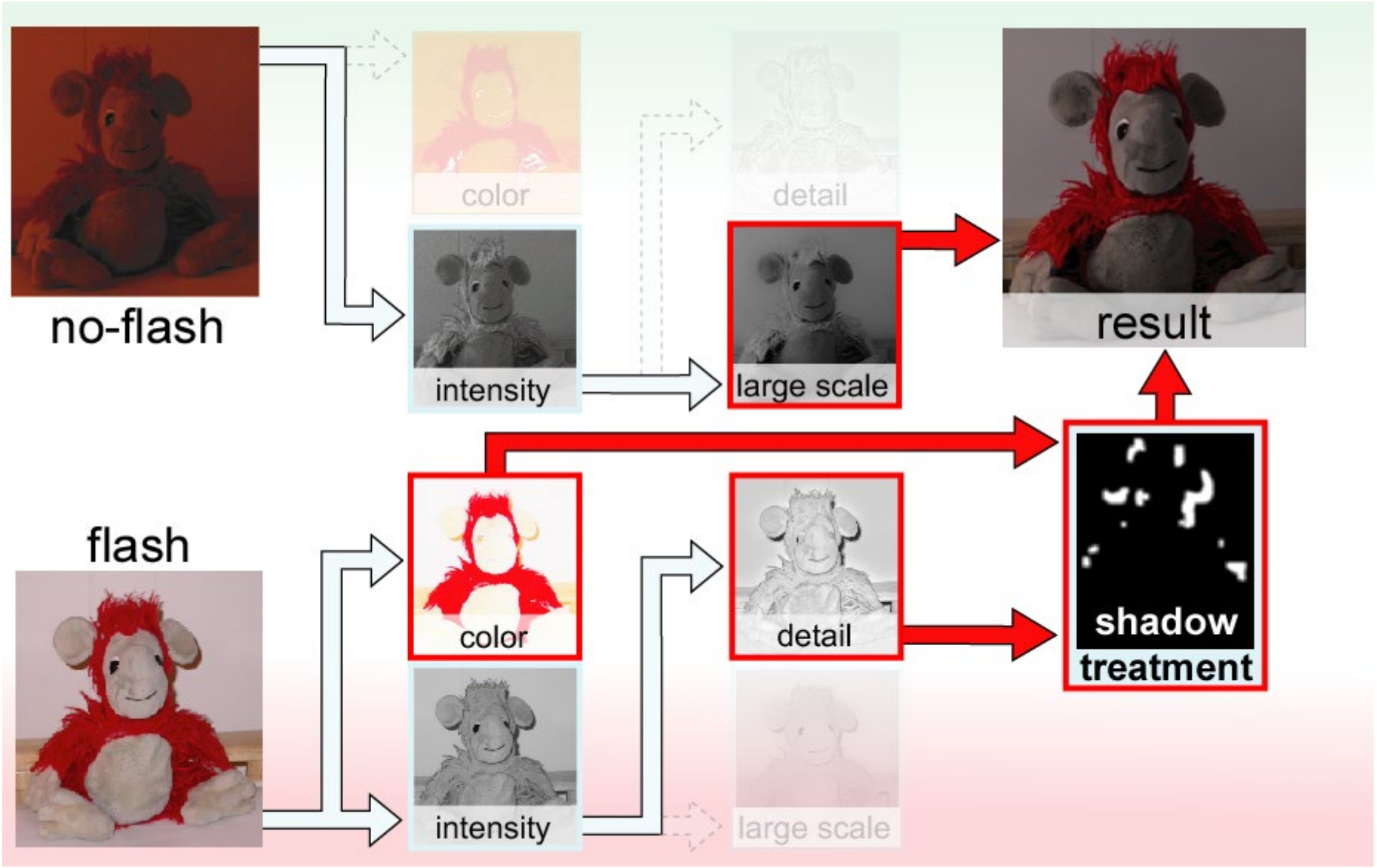
- One approach: Blend the two photos



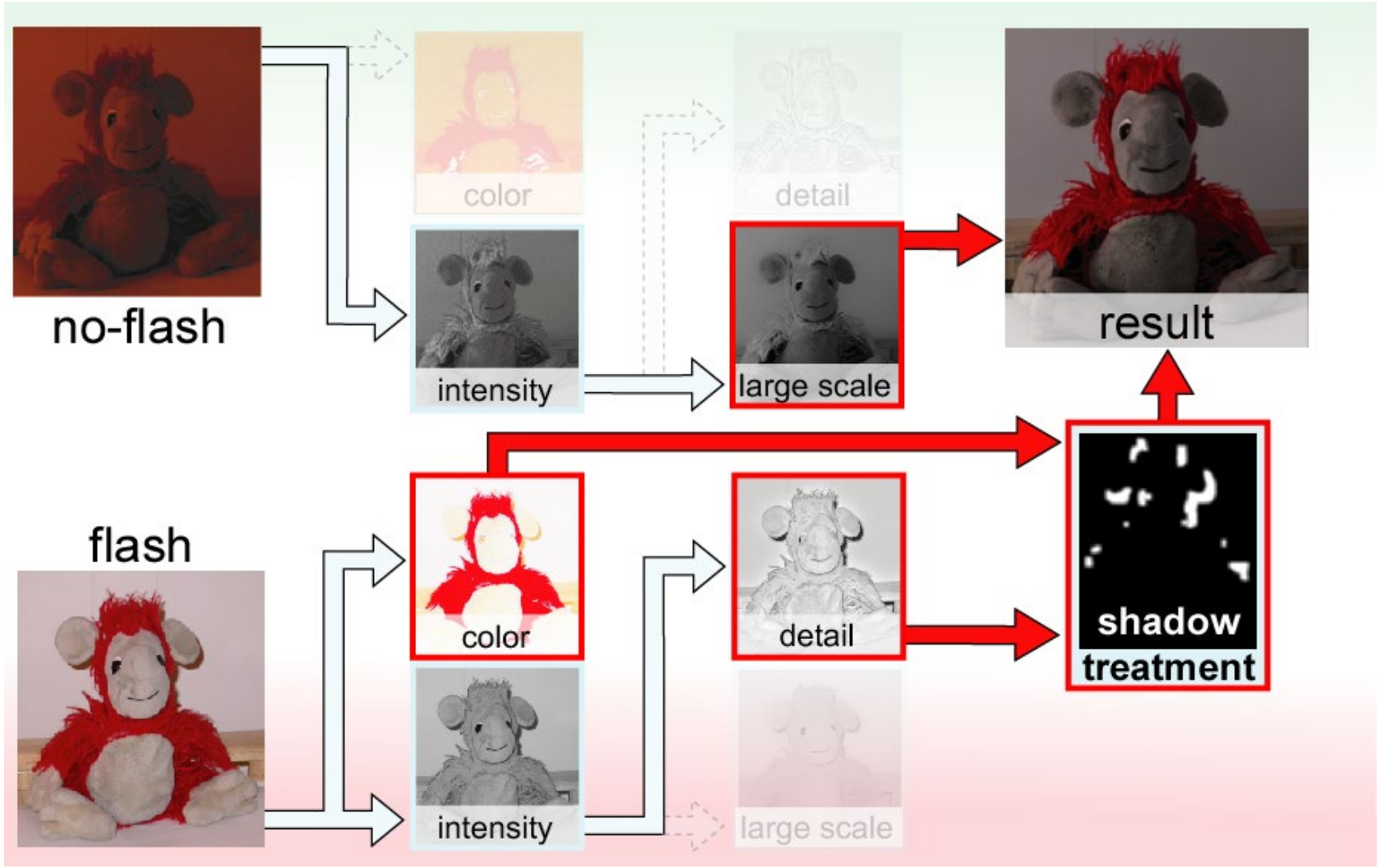
Main Idea


no-flash
lighting
result
flash
color
detail
shadow
treatment

Structure

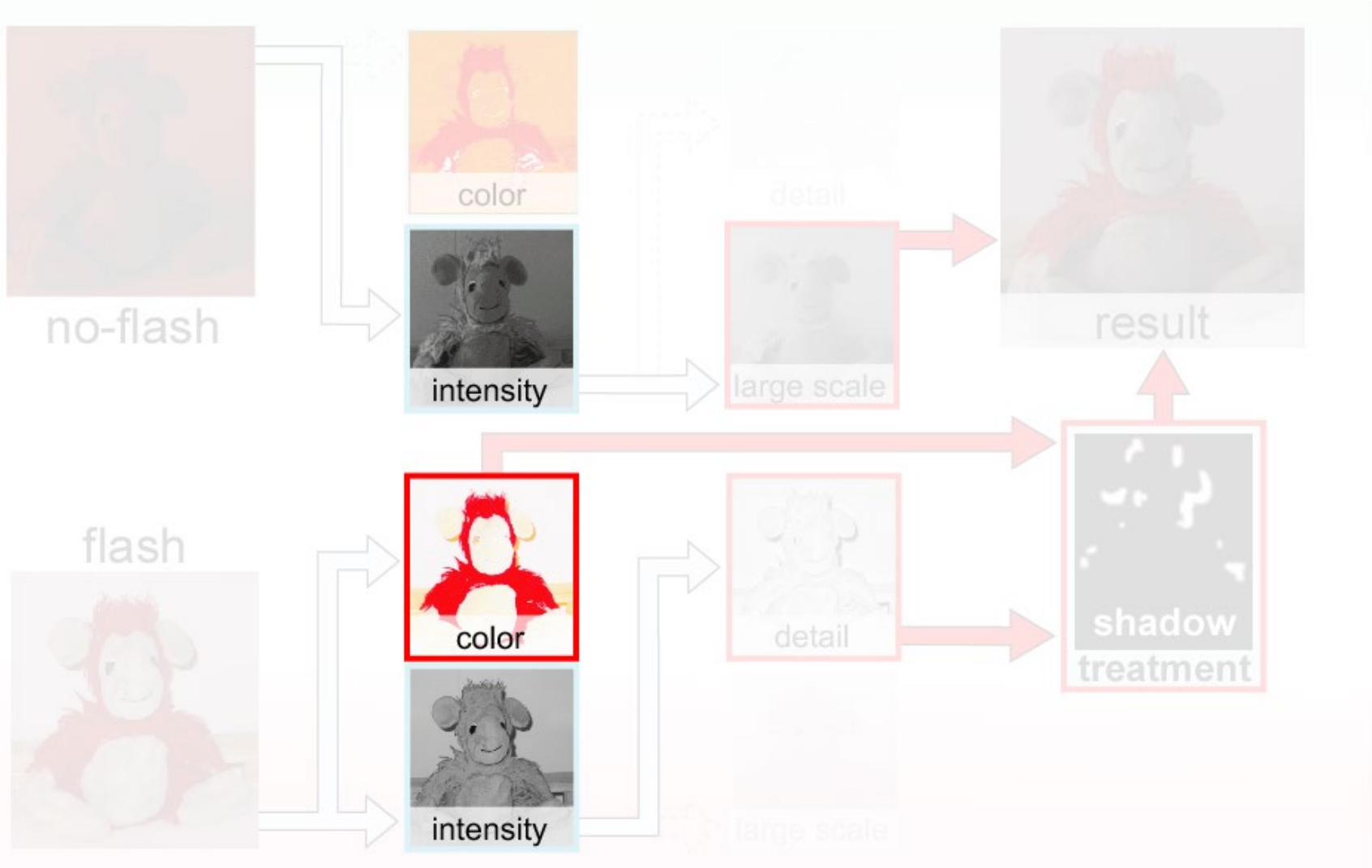

no-flash
color
detail
intensity
large scale
result
flash
color
detail
intensity
large scale
shadow
treatment

Structure


no-flash
color
detail
intensity
large scale
result
flash
color
detail
intensity
large scale
shadow
treatment

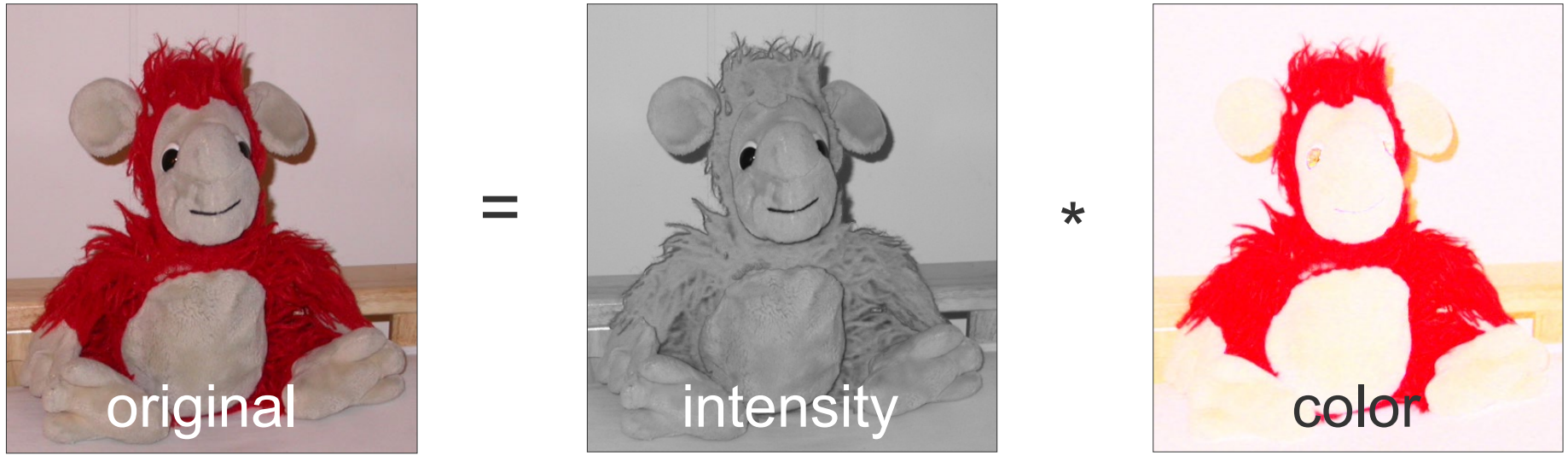
Intensity-Color Decomposition

Decomposition

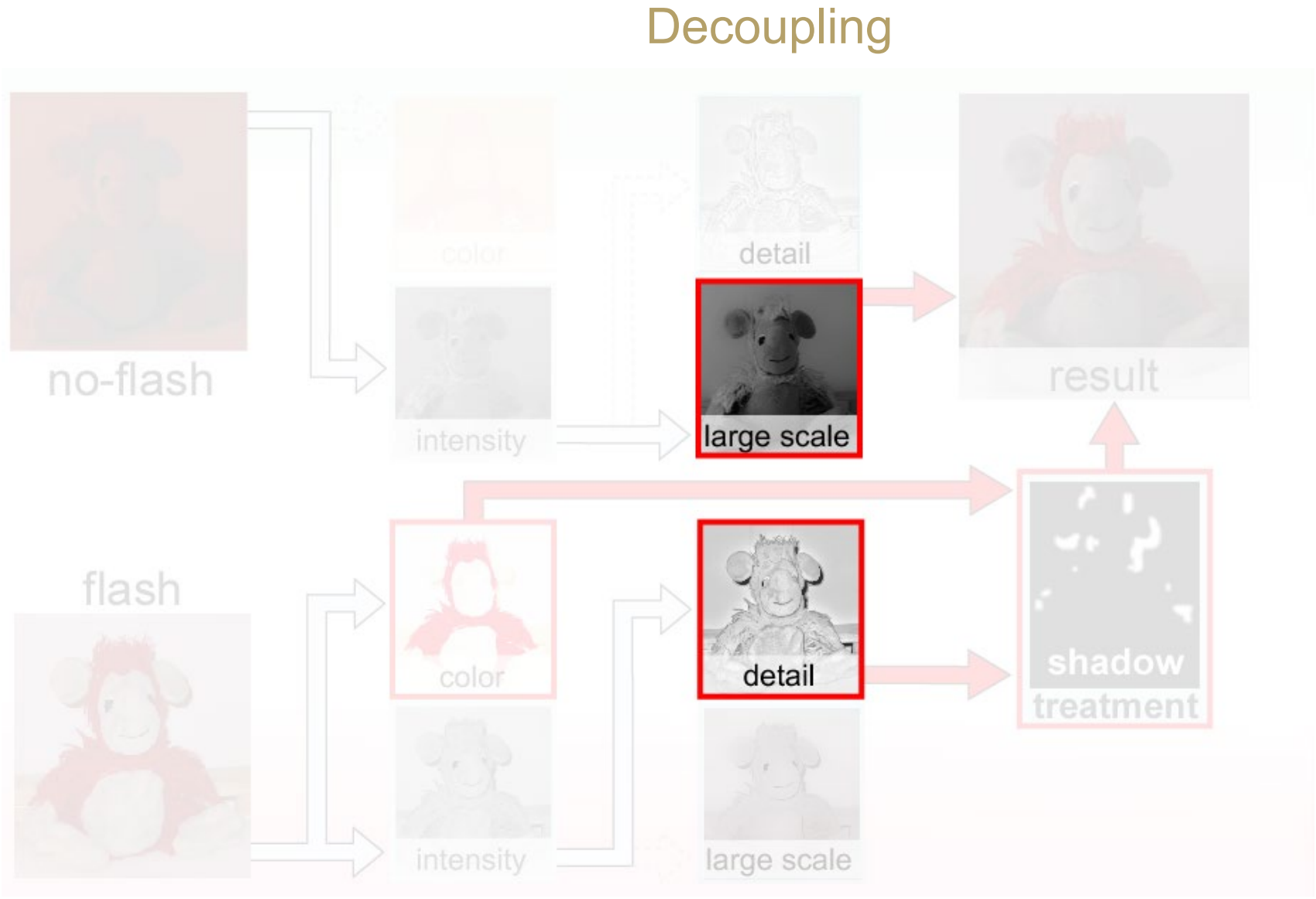


Decomposition

- Color / Intensity:



Frequency Decoupling



Decoupling

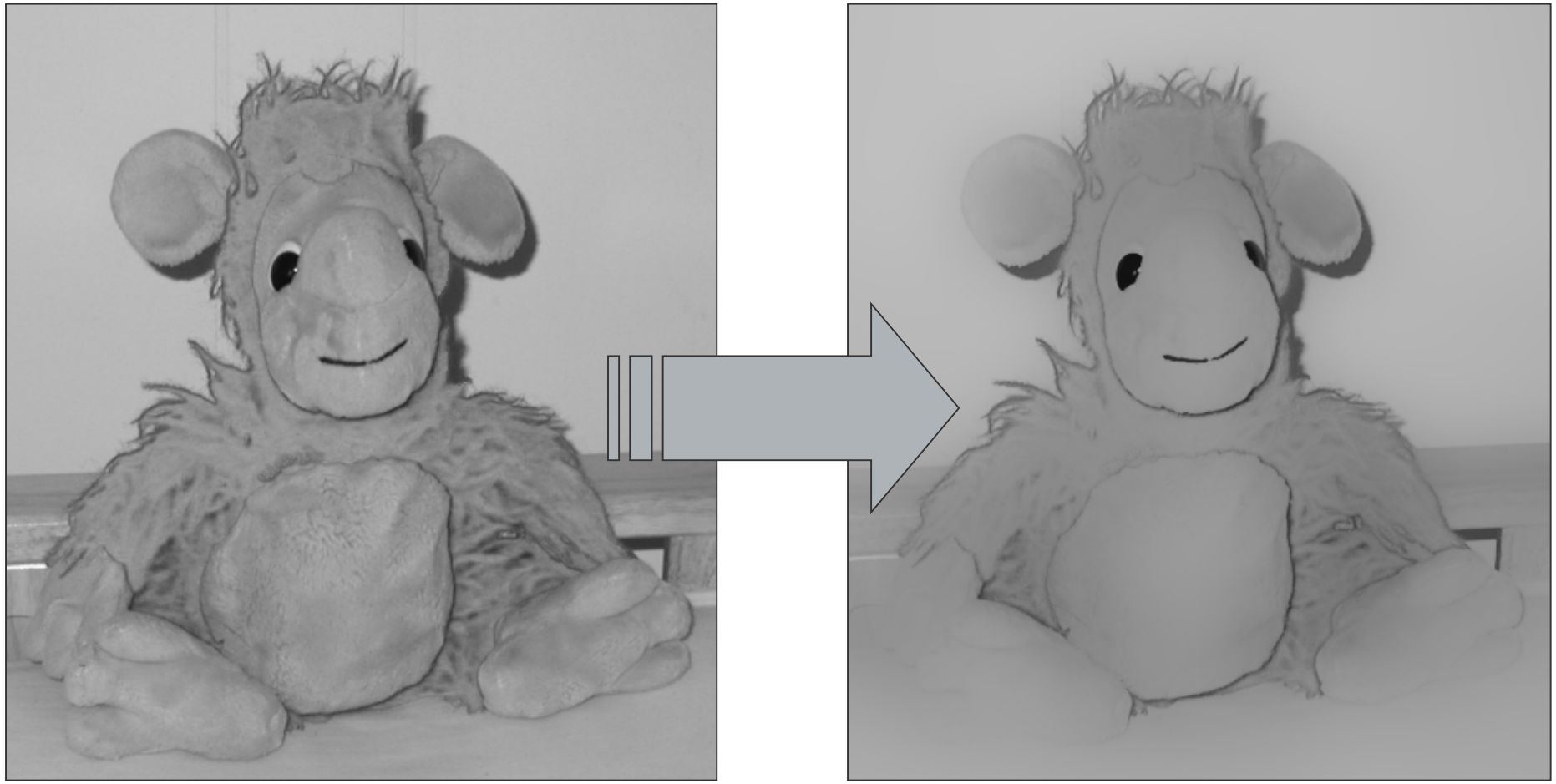
- Lighting : Large-scale variation
- Texture : Small-scale variation

Lighting

Texture

Large-scale Layer

- Bilateral filter



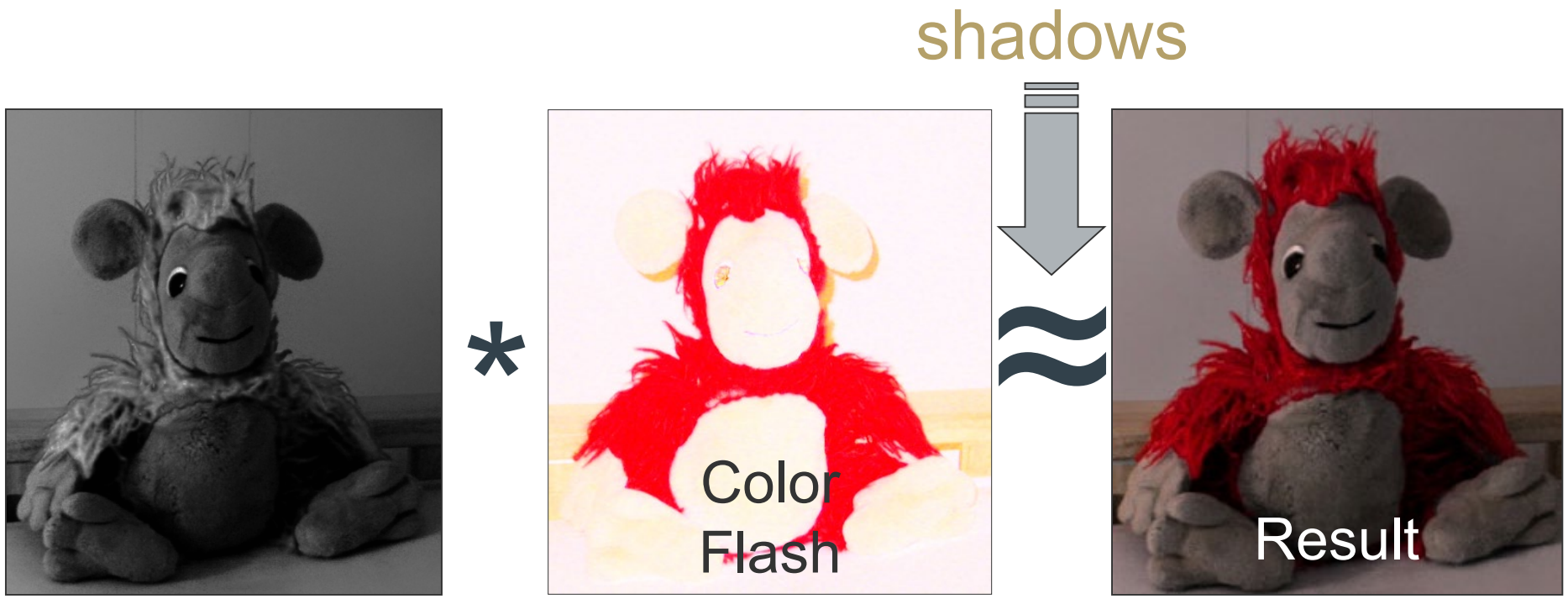
Detail Layer

Recombination: Large scale * Detail = Intensity

Recombination

Recombination: Large scale * Detail = Intensity

Recombination



Recombination: Intensity * Color = Original

Results

No-flash

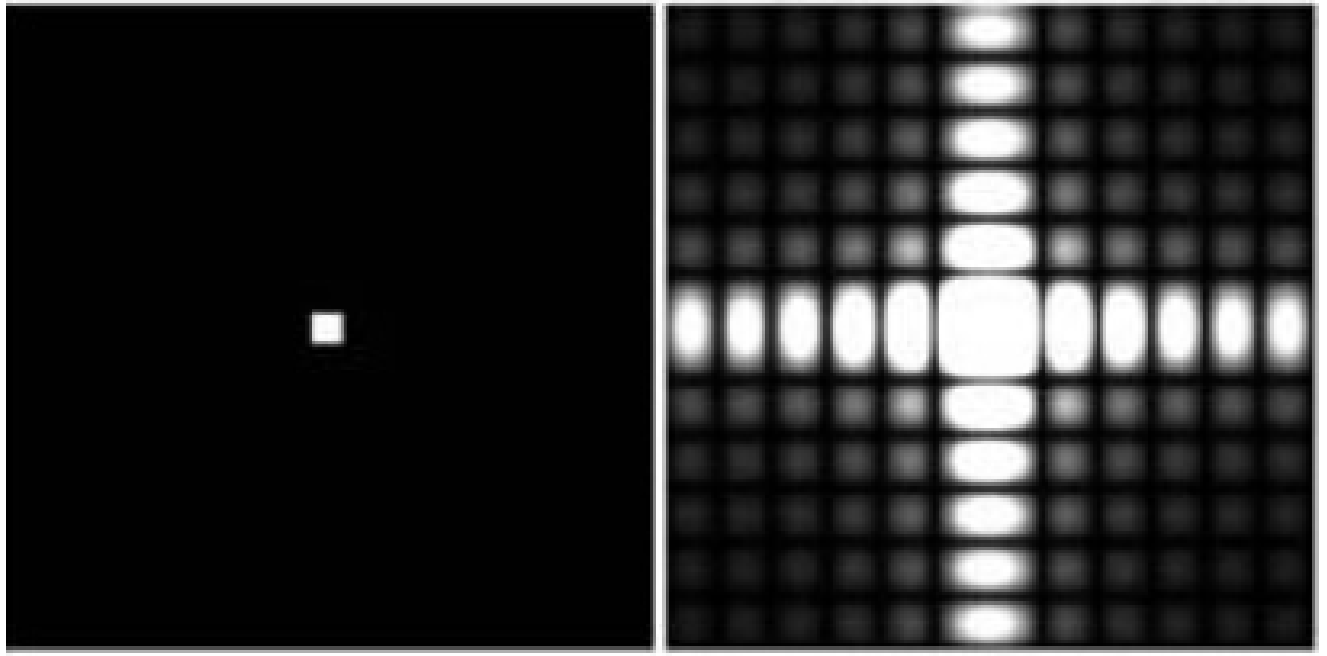
Flash

Result

Wavelets

Motivation

- Fourier or DCT perform a global transformation into the frequency domain
- Image areas where there is low/high frequency content are not distinguished



- This example a single spot in the spatial domain influences the entire frequency spectrum

Motivation (2)

Disadvantage of Fourier/DCT:
Basis function with global support in spatial domain

- Not very efficient in representing the signal
- Loss of any spatial information
- Operations in the frequency domain cannot be confined to specific region in spatial domain

Motivation (3)

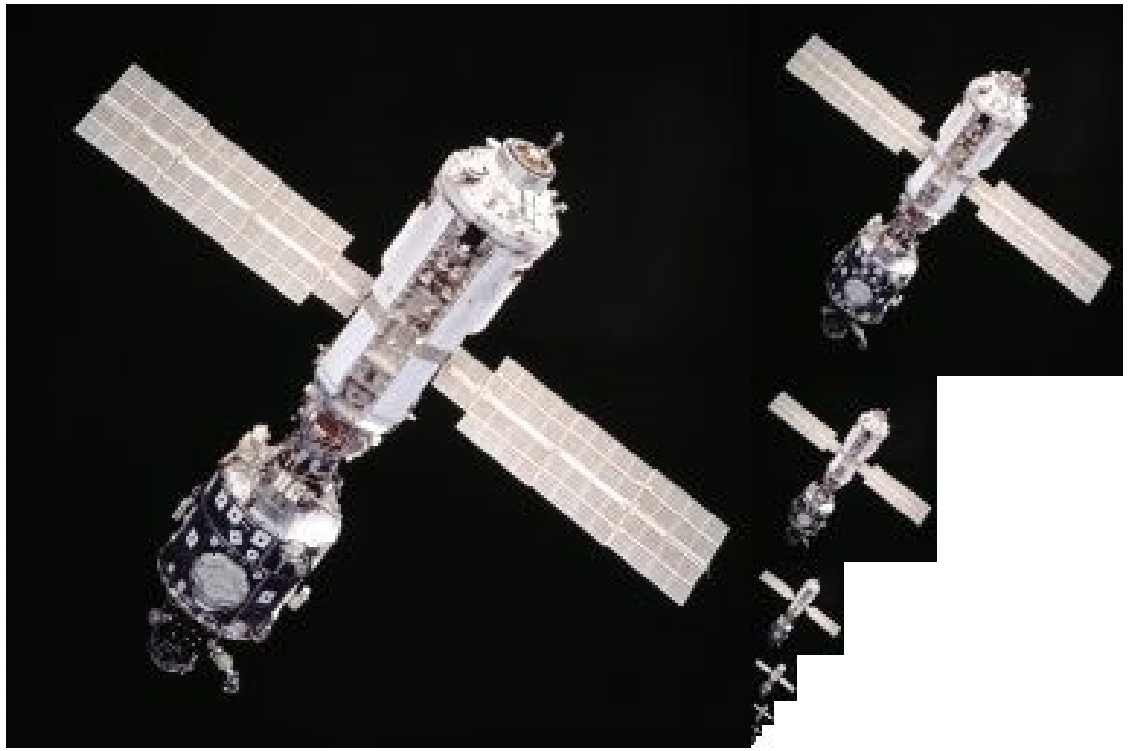
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

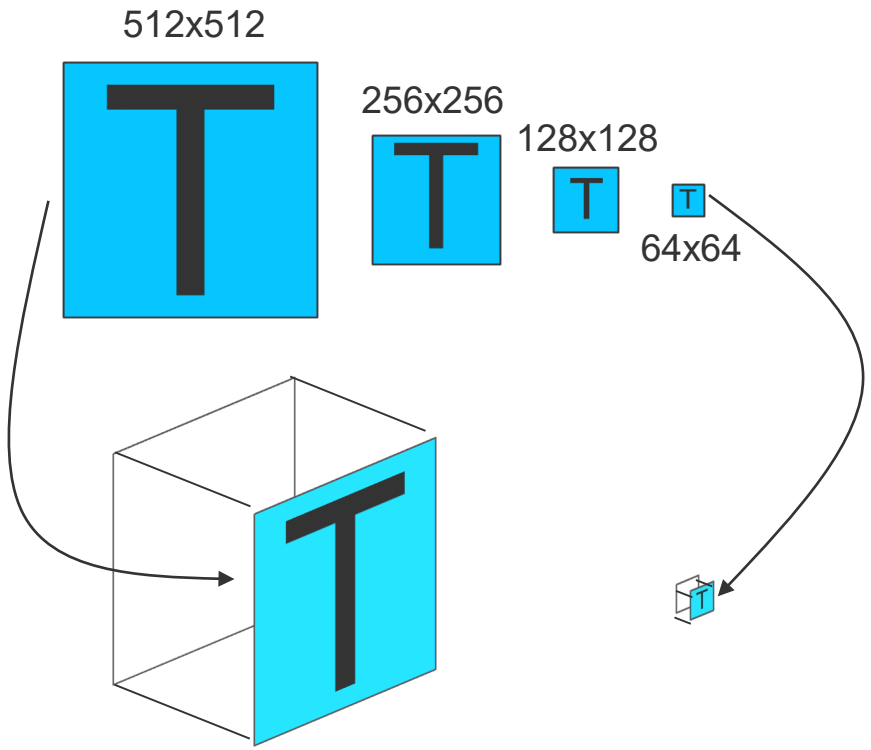
- Fourier/DCT:
  - Represents the signal/image as linear combination of basis functions with varying frequency

- Wavelets:
  - Representation as linear combination of basis function with varying frequency (similar to Fourier/DCT)
  - Wavelet basis function with ***spatially localized support***

MipMapping

- Texture available in multiple resolutions
  - Pre-processing step
- Rendering: select appropriate texture resolution
  - Selection is usually per pixel !!
  - Texel size(n) < extent of pixel footprint < texel size(n+1)





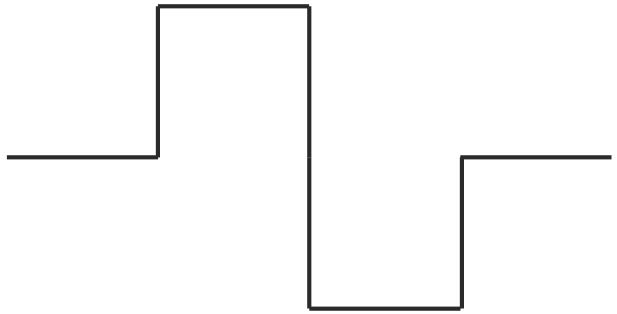
Haar Wavelet Basis

- mother scaling function

$$\phi(x) := \begin{cases} 1, & for\ 0 \le x < 1 \\ 0, & otherwise \end{cases}$$

- mother wavelet function

$$\psi(x) := \begin{cases} 1, & for\ 0 \le x < 1/2 \\ -1, & for\ 1/2 \le x < 1 \\ 0, & otherwise. \end{cases}$$



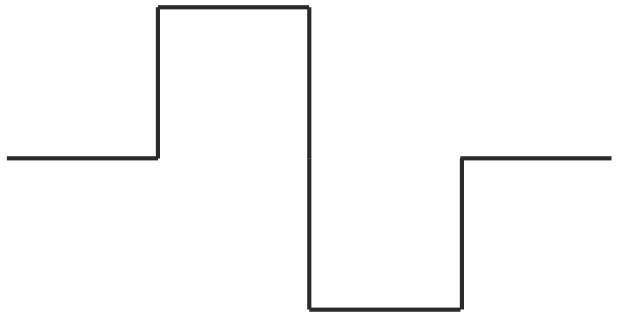
Haar Wavelet Basis

- mother scaling function

$$\phi(x) := \begin{cases} 1, & for\ 0 \le x < 1 \\ 0, & otherwise \end{cases}$$

- mother wavelet function

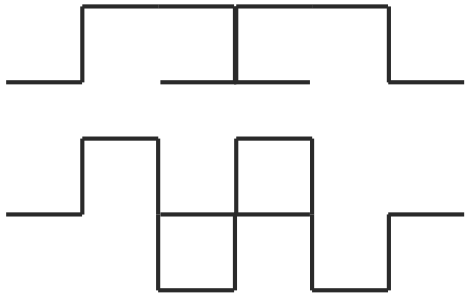
$$\psi(x) := \begin{cases} 1, & for\ 0 \le x < 1/2 \\ -1, & for\ 1/2 \le x < 1 \\ 0, & otherwise. \end{cases}$$



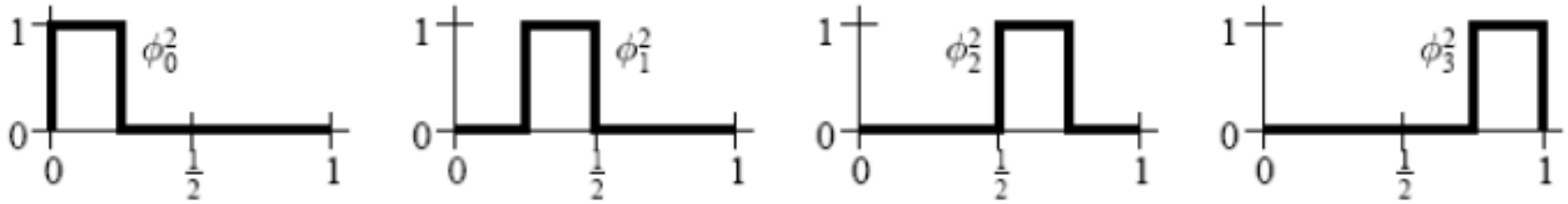
- Normalized scaling and wavelet basis functions

$$\phi_t^l(x) := 2^{l/2}\phi(2^l x - t)$$

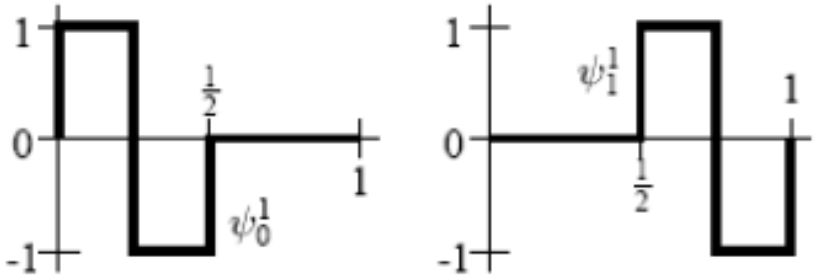
$$\psi_t^l(x) := 2^{l/2}\psi(2^l x - t)$$



Haar Basis Functions



Scaling Functions for V^2



Wavelets for W^1

Wavelet Representation

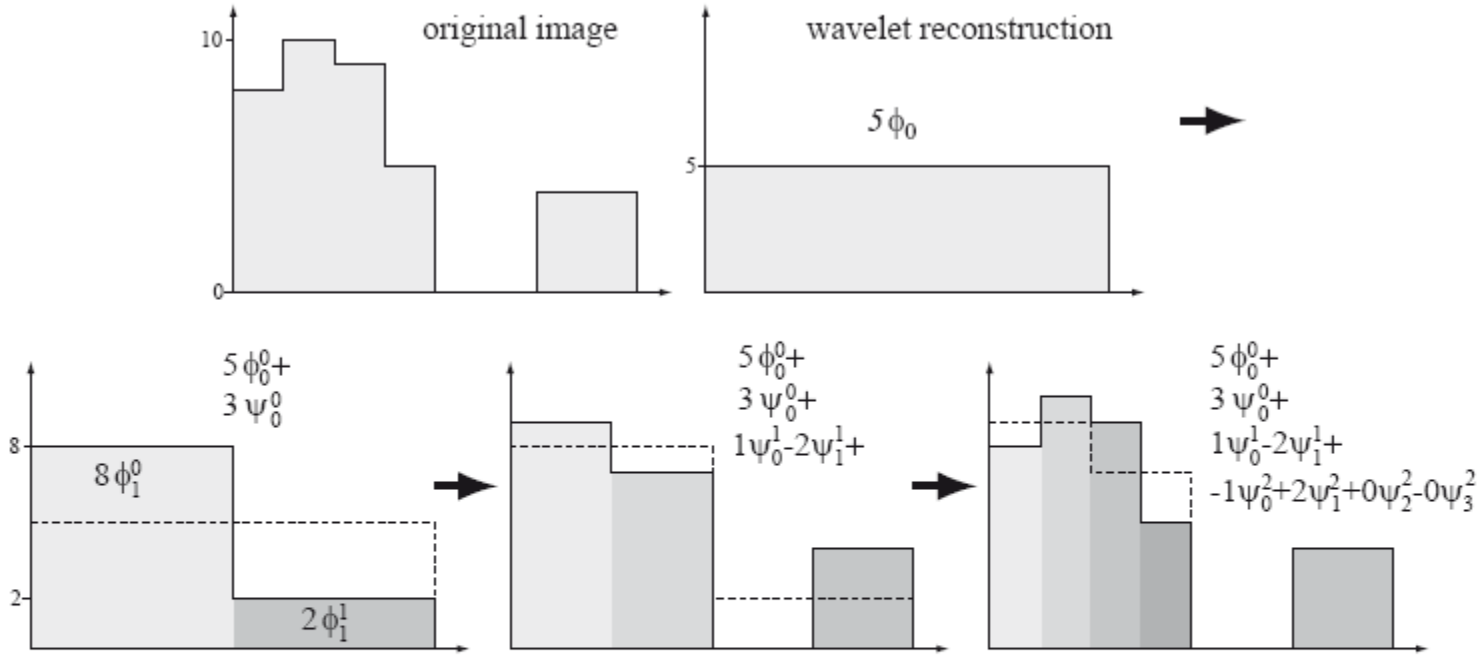
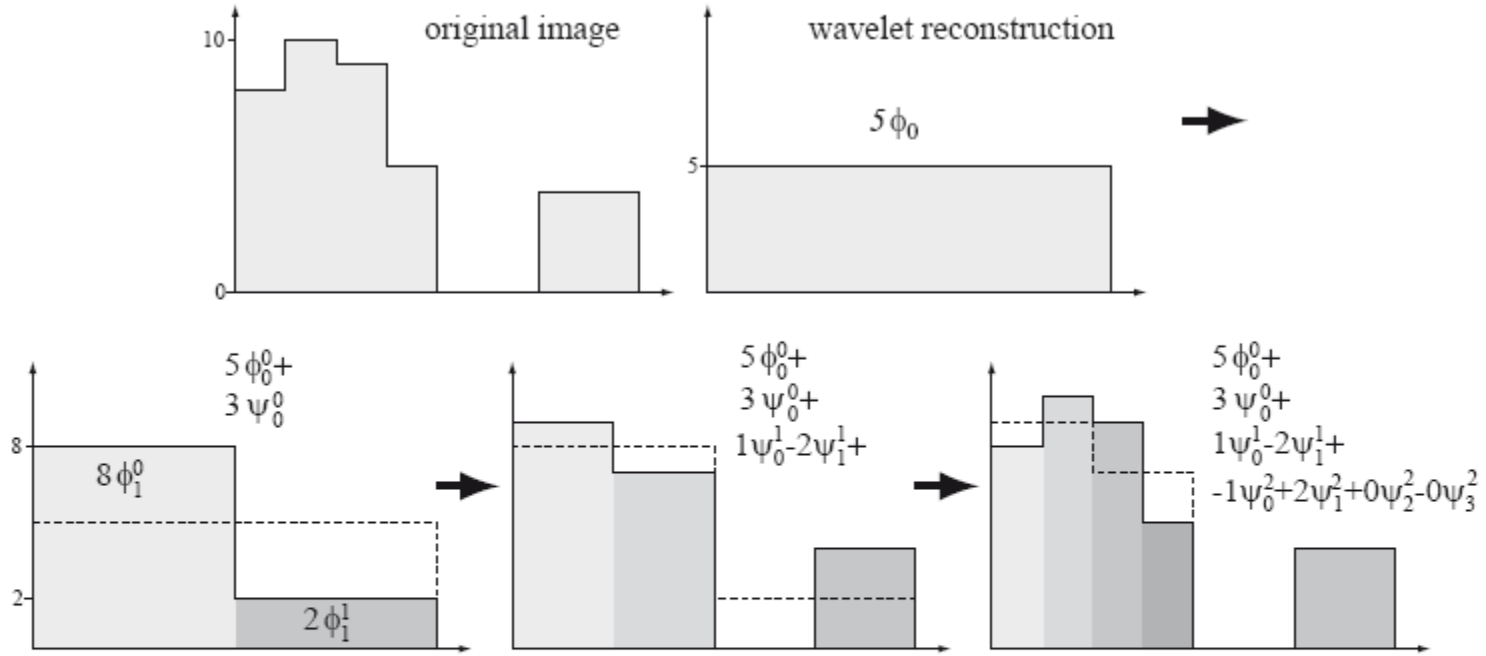


Figure 3: *The 1D image (upper, left) is:* $[8, 10, 9, 5, 0, 0, 4, 4]$, *and its unnormalized (used here because it is simpler to display) Haar representation is:* $[5, 3, 1, -2, -1, 2, 0, 0]$. *The image is then reconstructed one level at a time as follows:* $[5] \rightarrow [5+3, 5-3] = [8, 2] \rightarrow [8+1, 8-1, 2-2, 2+2] = [9, 7, 0, 4]$ *and so on.*

Wavelet Representation



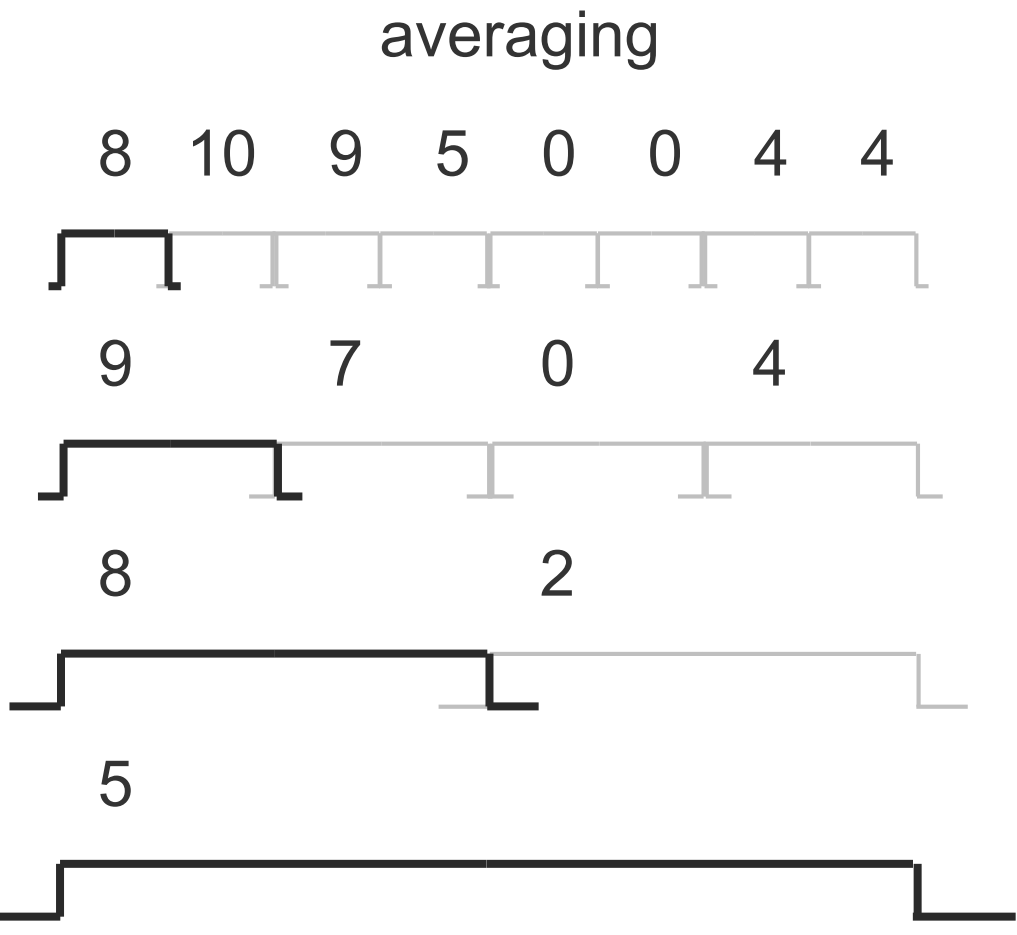
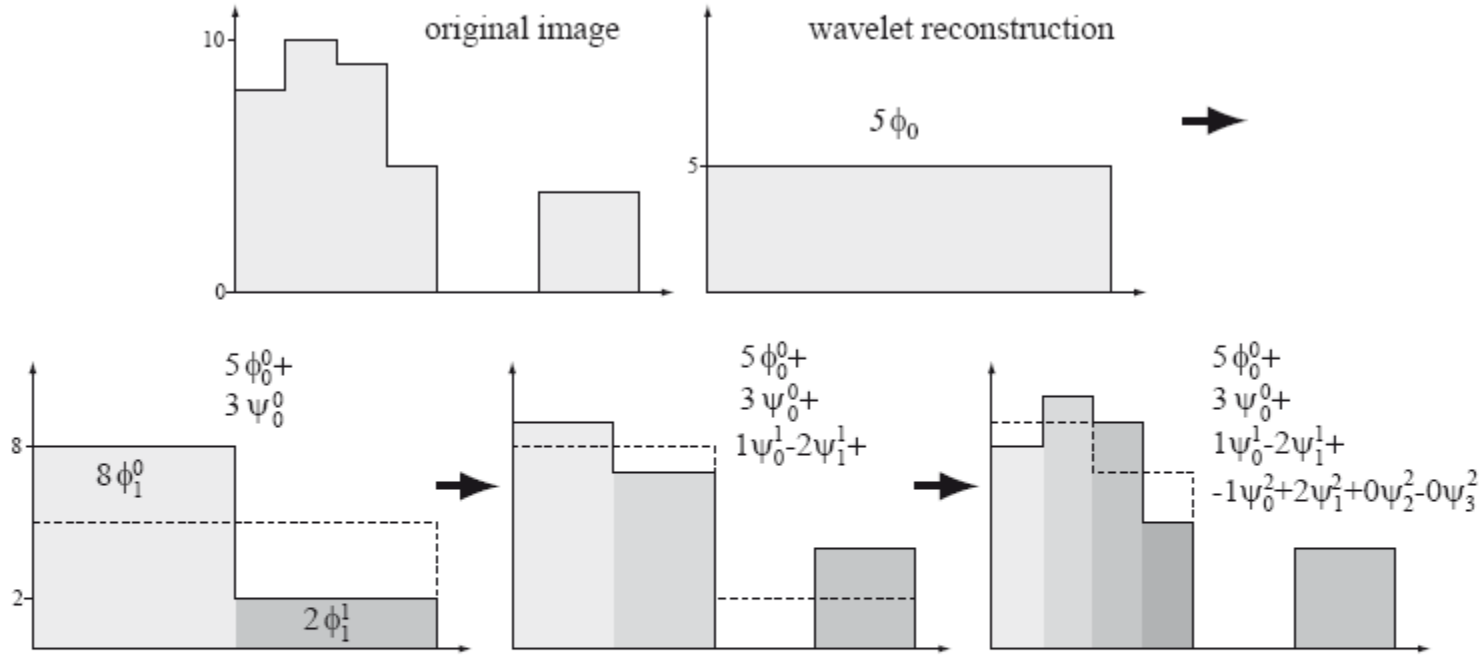


Figure 3: *The 1D image (upper, left) is:* $[8, 10, 9, 5, 0, 0, 4, 4]$, *and its unnormalized (used here because it is simpler to display) Haar representation is:* $[5, 3, 1, -2, -1, 2, 0, 0]$. *The image is then reconstructed one level at a time as follows:* $[5] \rightarrow [5+3, 5-3] = [8, 2] \rightarrow [8+1, 8-1, 2-2, 2+2] = [9, 7, 0, 4]$ *and so on.*

Wavelet Representation



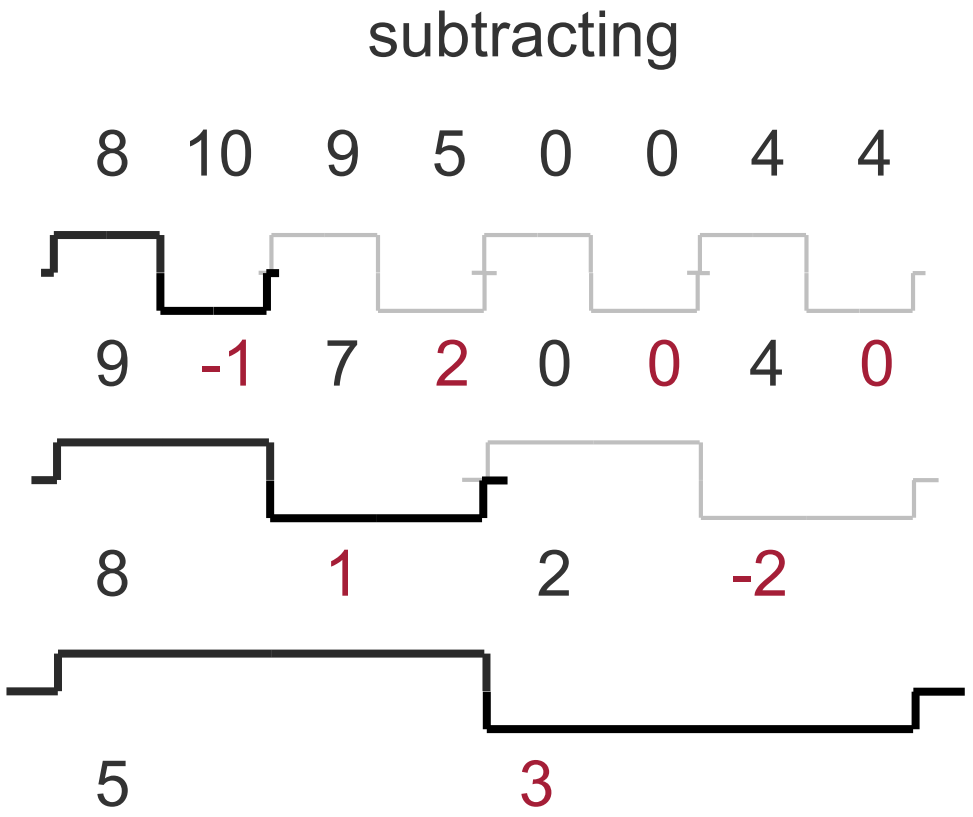
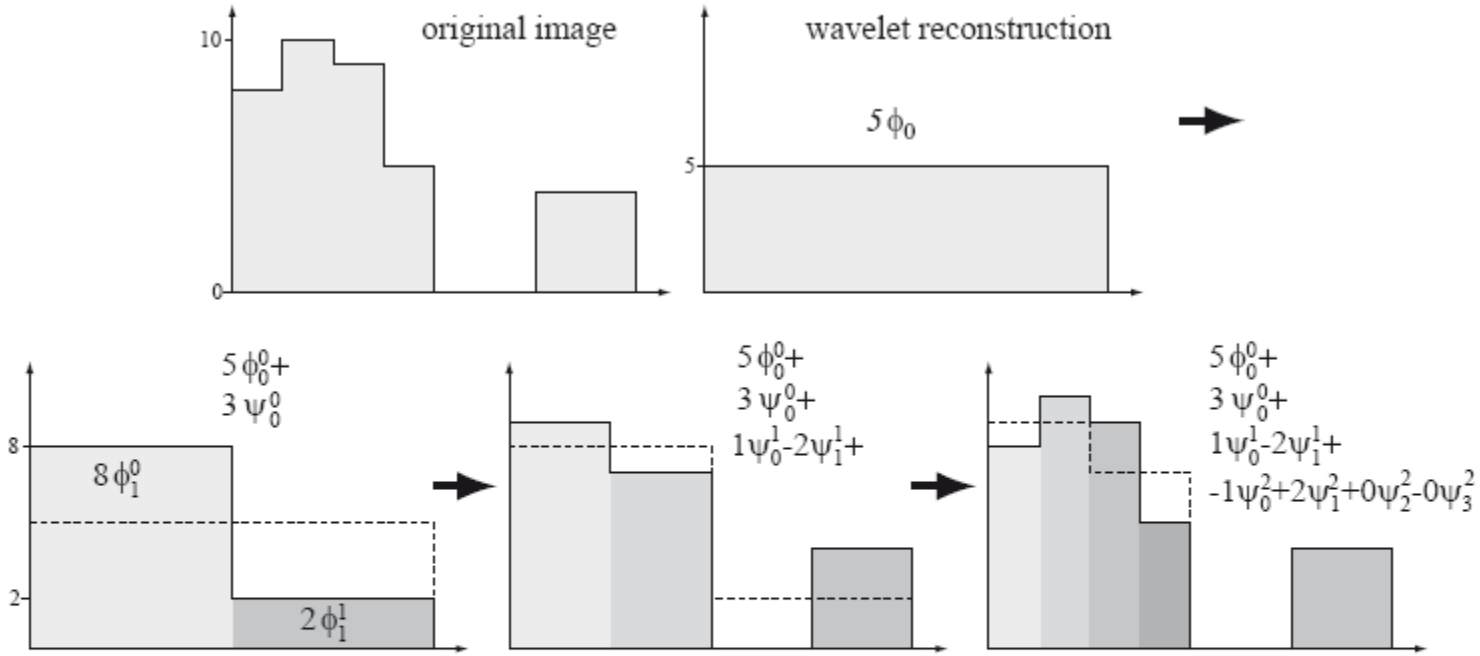


Figure 3: *The 1D image (upper, left) is:* $[8, 10, 9, 5, 0, 0, 4, 4]$, *and its unnormalized (used here because it is simpler to display) Haar representation is:* $[5, 3, 1, -2, -1, 2, 0, 0]$. *The image is then reconstructed one level at a time as follows:* $[5] \rightarrow [5+3, 5-3] = [8, 2] \rightarrow [8+1, 8-1, 2-2, 2+2] = [9, 7, 0, 4]$ *and so on.*

Wavelet Representation – Reconstruction



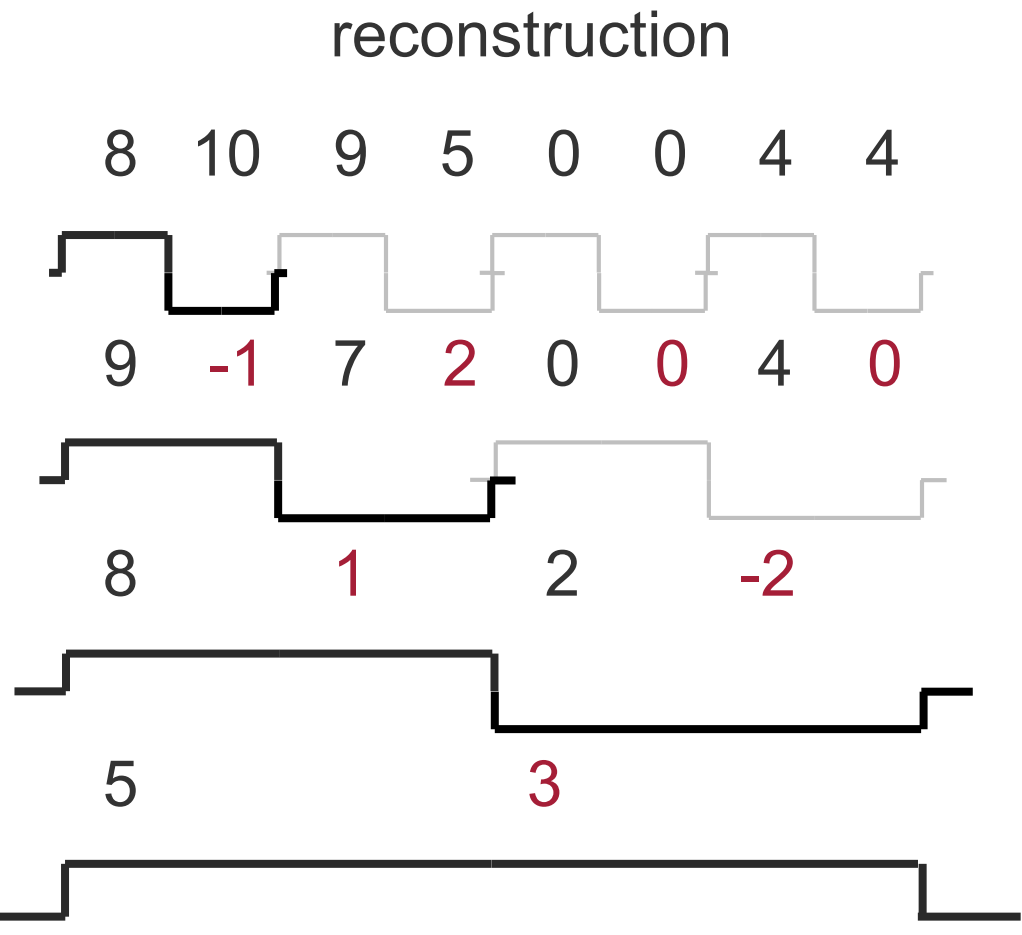
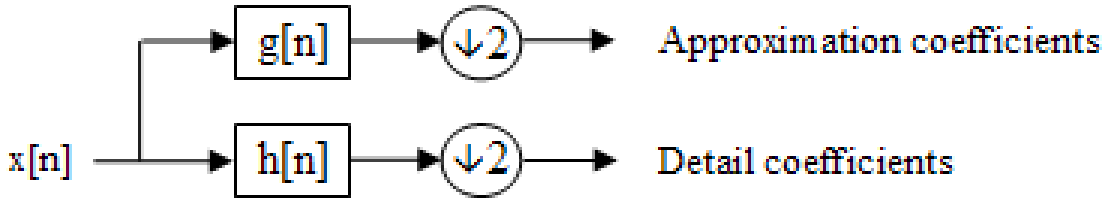


Figure 3: *The 1D image (upper, left) is:* $[8, 10, 9, 5, 0, 0, 4, 4]$, *and its unnormalized (used here because it is simpler to display) Haar representation is:* $[5, 3, 1, -2, -1, 2, 0, 0]$. *The image is then reconstructed one level at a time as follows:* $[5] \rightarrow [5+3, 5-3] = [8, 2] \rightarrow [8+1, 8-1, 2-2, 2+2] = [9, 7, 0, 4]$ *and so on.*

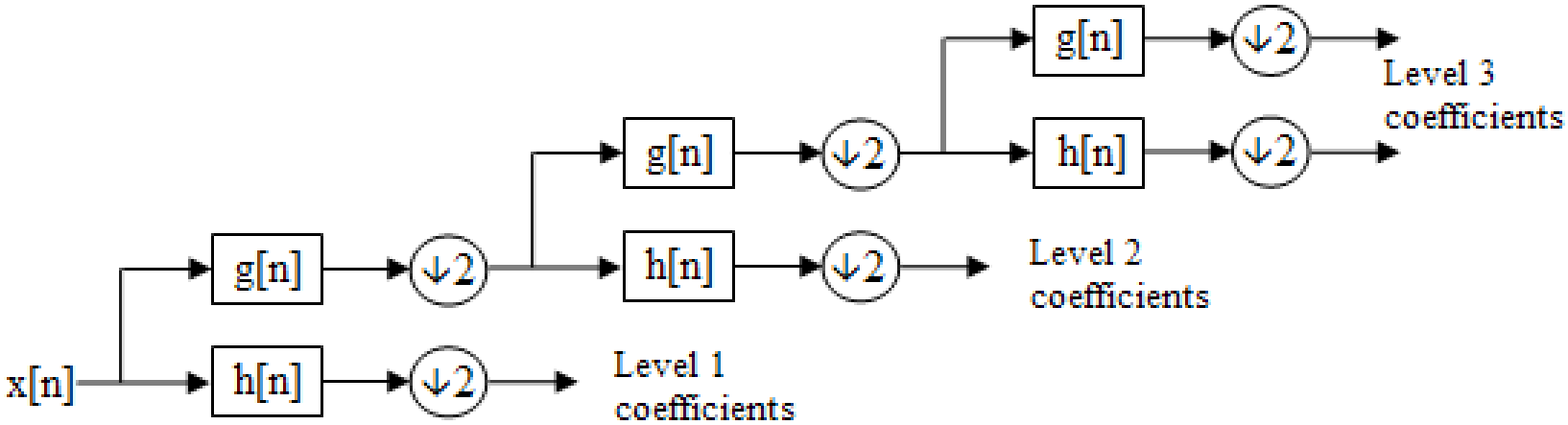
Fast Wavelet Transform

- Downsampling operator e.g.
- selecting every other sample high-pass and low-pass filter

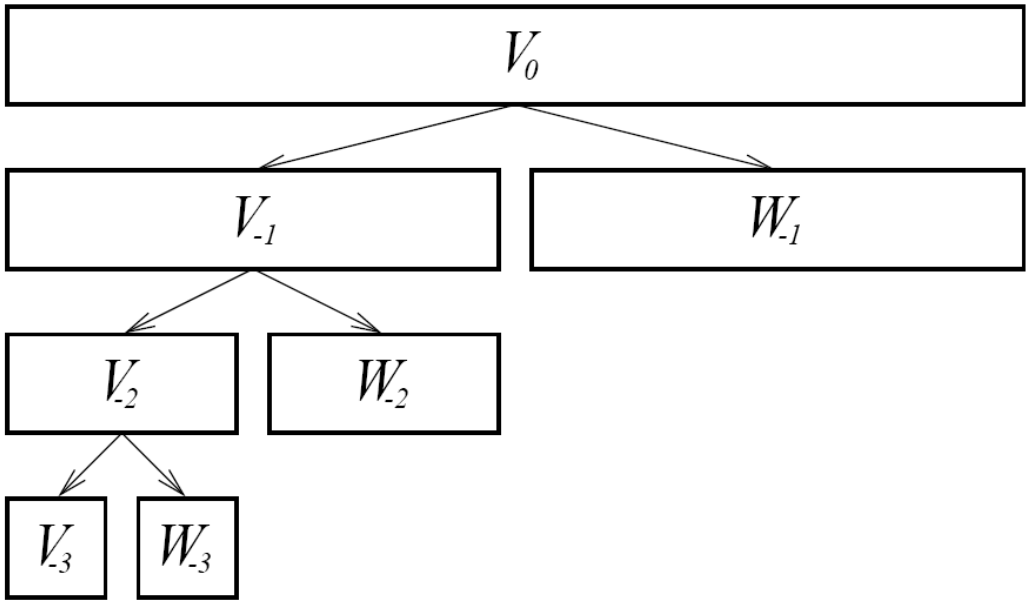
- Single level transform

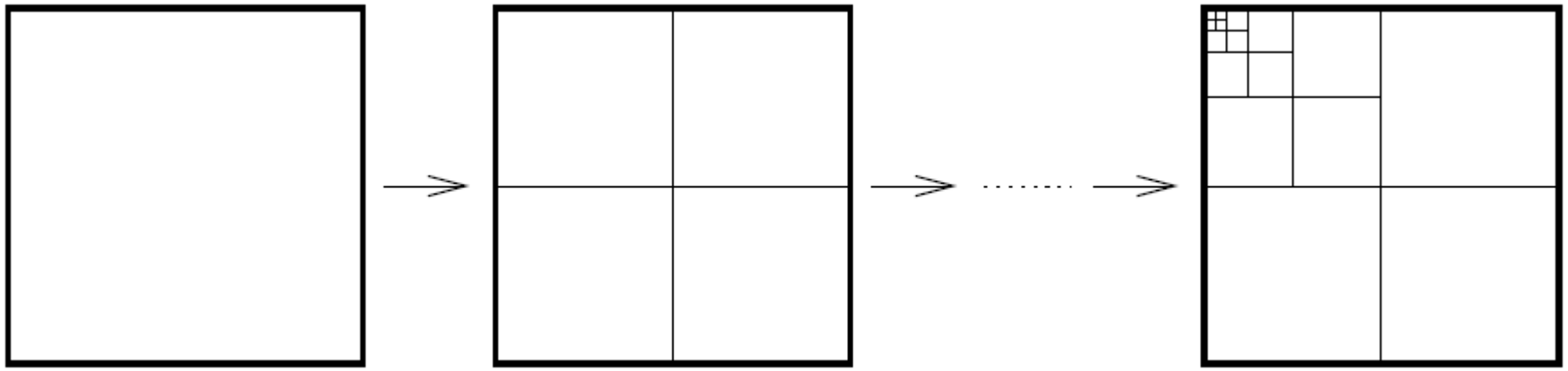


- Multi Resolution Analysis

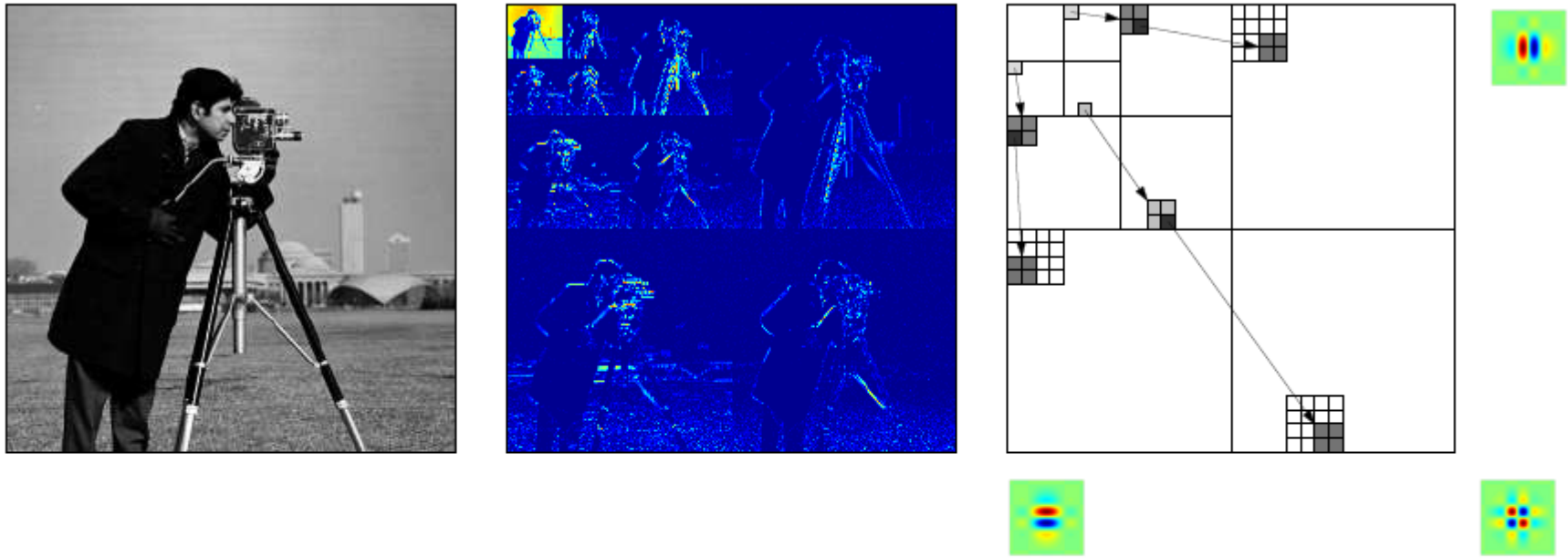


Coefficients



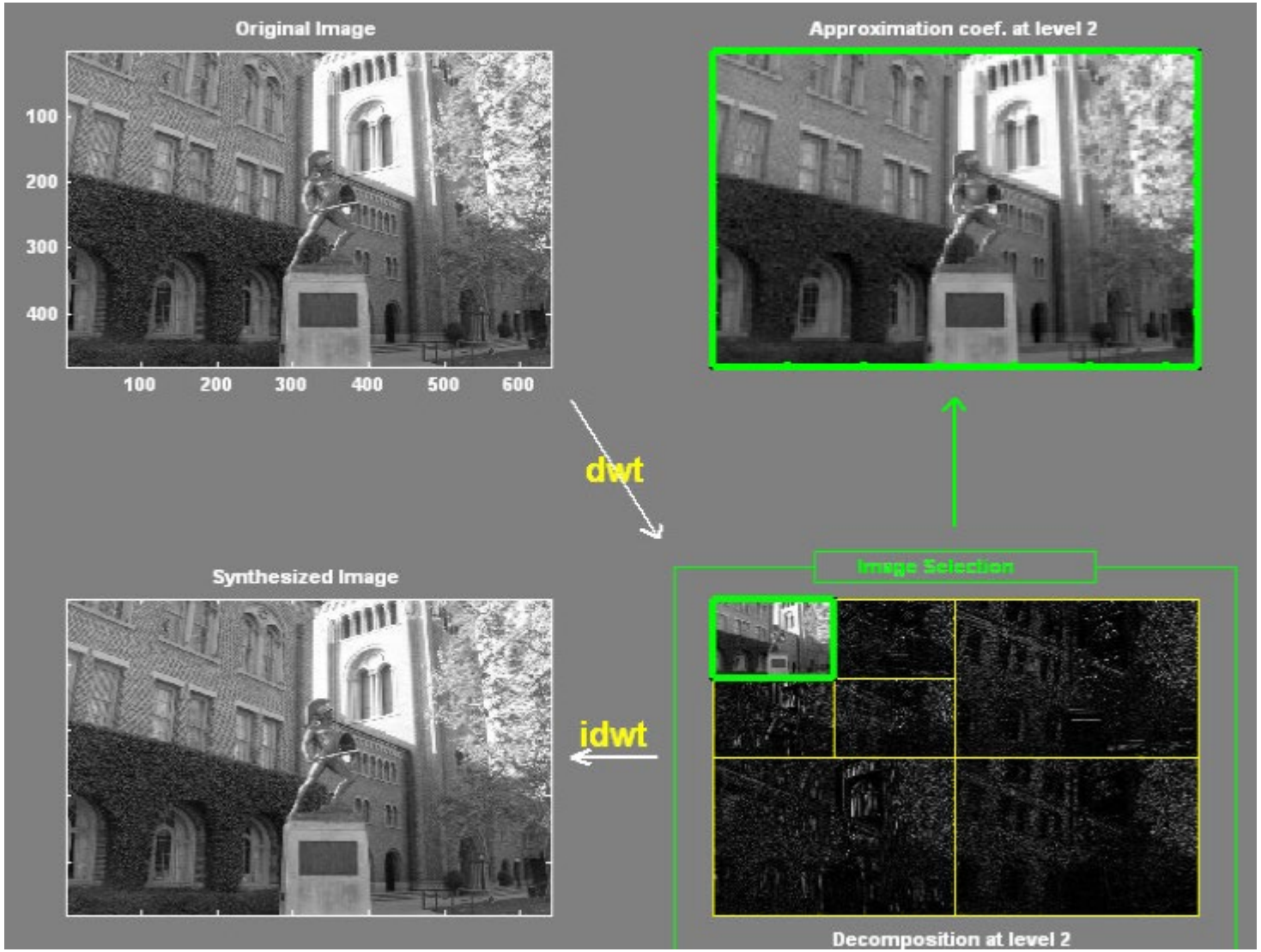


2D Wavelets

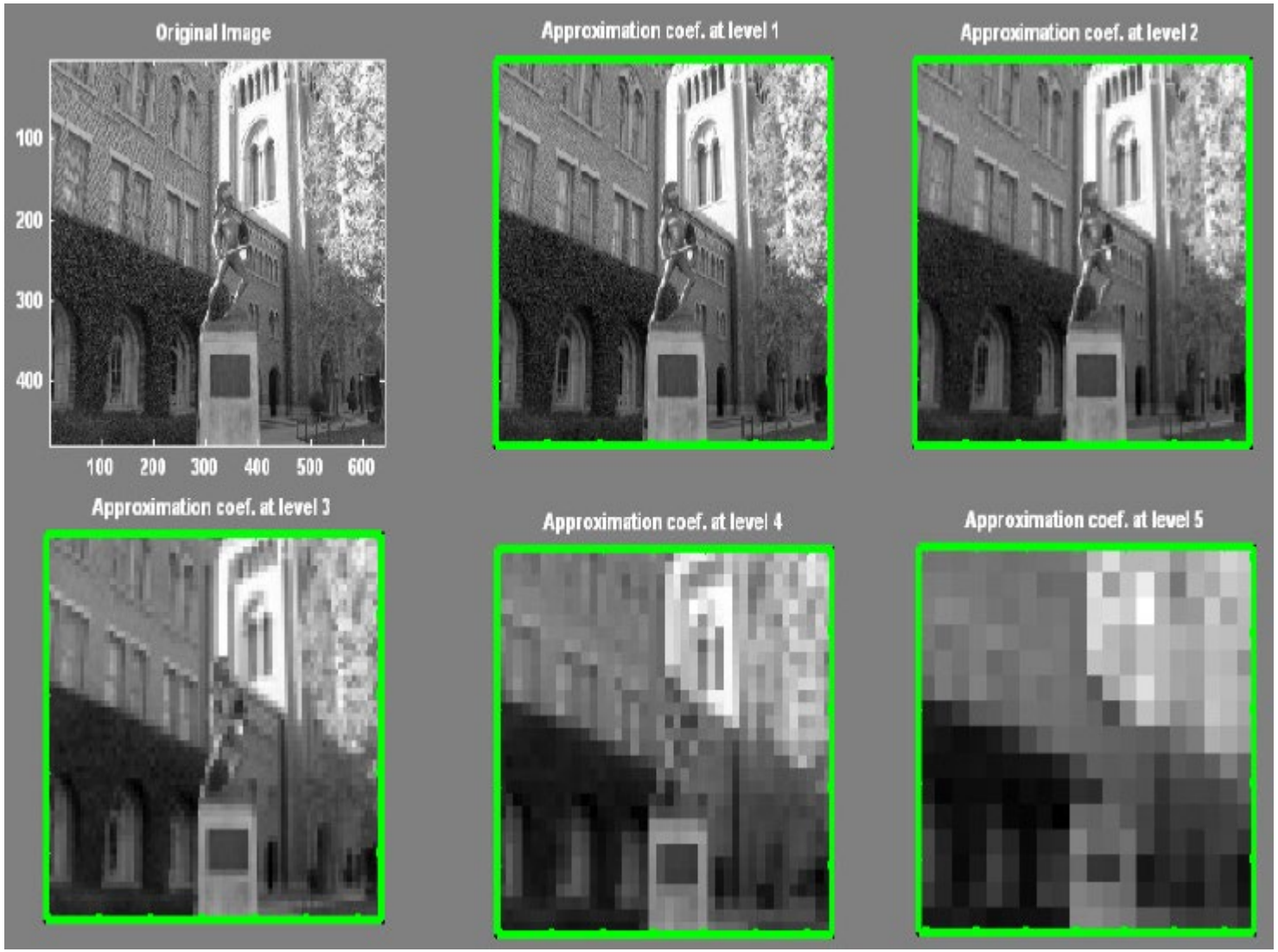


- Sparse structure: few large coeffs, many small coeffs
- Basis for JPEG2000 image compression standard
- Wavelet approximations: smooths regions great, edges much sharper
- *Fundamentally better than DCT for images with edges*

Example


Original Image
100
200
300
400
100
200
300
400
500
600
Approximation coef. at level 2
dwt
Image Selection
Synthesized Image
idwt
Decomposition at level 2

Example


Original Image
100
200
300
400
100 200 300 400 500 600
Approximation coef. at level 1
Approximation coef. at level 2
Approximation coef. at level 3
Approximation coef. at level 4
Approximation coef. at level 5

Questions

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- What happens if I leave out the detail coefficients of the last two levels?
- How many detail coefficients do you need to faithfully represent an image?

  10%, 25%, 50%, 75% or 100%?

Wavelet Representation

8	10	9	5	0	0	4	4
	-1		2		0		0
		1				-2	
5				3			

- For representing H(x) in Haar wavelet basis you need only
  - scaling coefficient for the first level
  - all detail coefficients

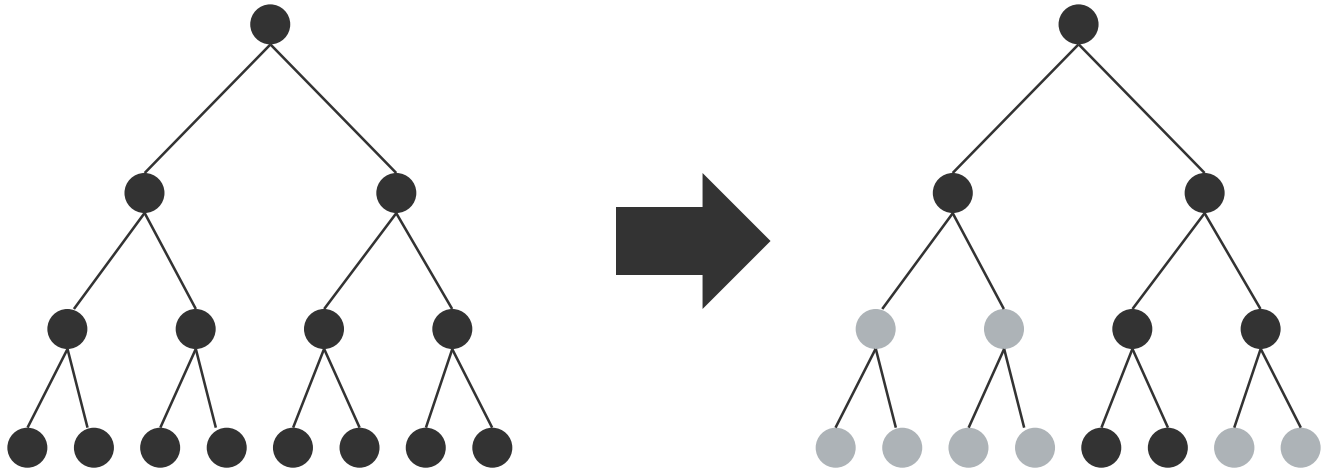
$$H(x) = H_{0,0}^{0}\phi_{0}^{0} + \sum_{l}\sum_{t} H_{t,1}^{l}\psi_{t}^{l} = \sum_{i} H_{i}\Psi_{i}$$

- For a sampled function a lossless representation is achieved if #samples==#wavelet coefficients

- Compression by setting small wavelet coefficients to zero

Sparse Representation

- First measure entire matrix
- Then compress by determining the sparse tree
  - since all information is given you are on the save side



Compression - Shrinkage

- The goal of compression is to express an initial set of data using some smaller set of data, either with or without loss of information.

$$f(x) = \sum_{i=1}^{m} c_i u_i(x) \quad \Longrightarrow \quad \tilde{f}(x) = \sum_{i=1}^{\tilde{m}} \tilde{c}_i \tilde{u}_i(x)$$

- For orthonormal basis the square of the *L2 error in this* approximation is

$$\left\| f(x) - \tilde{f}(x) \right\|_2^2 = \sum_{i=\tilde{m}+1}^{m} (c_i)^2$$

- Compression – set all coefficients <t to zero!

$$\text{such that } \tilde{m} < m \text{ and } \left\| \tilde{f}(x) - f(x) \right\| < \varepsilon \text{ for some norm}$$

Wavelet image compression

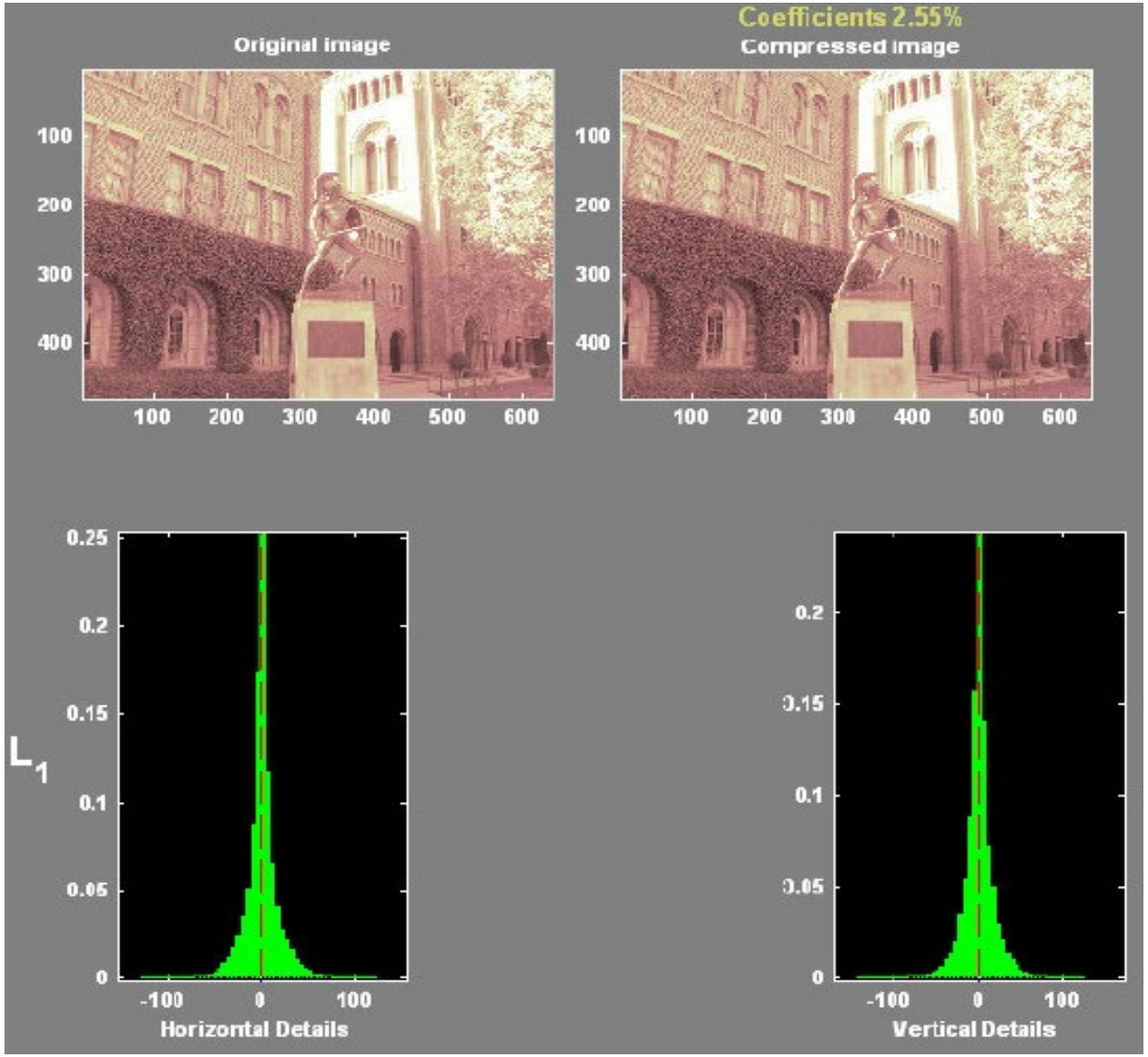
1. Compute coefficients *c1, …, cm representing an image in a* normalized two-dimensional Haar basis.

2. Sort the coefficients in order of decreasing magnitude to produce the sequence *c(1), … , c(m).*

3. Find the smallest $\tilde{m}$ so that

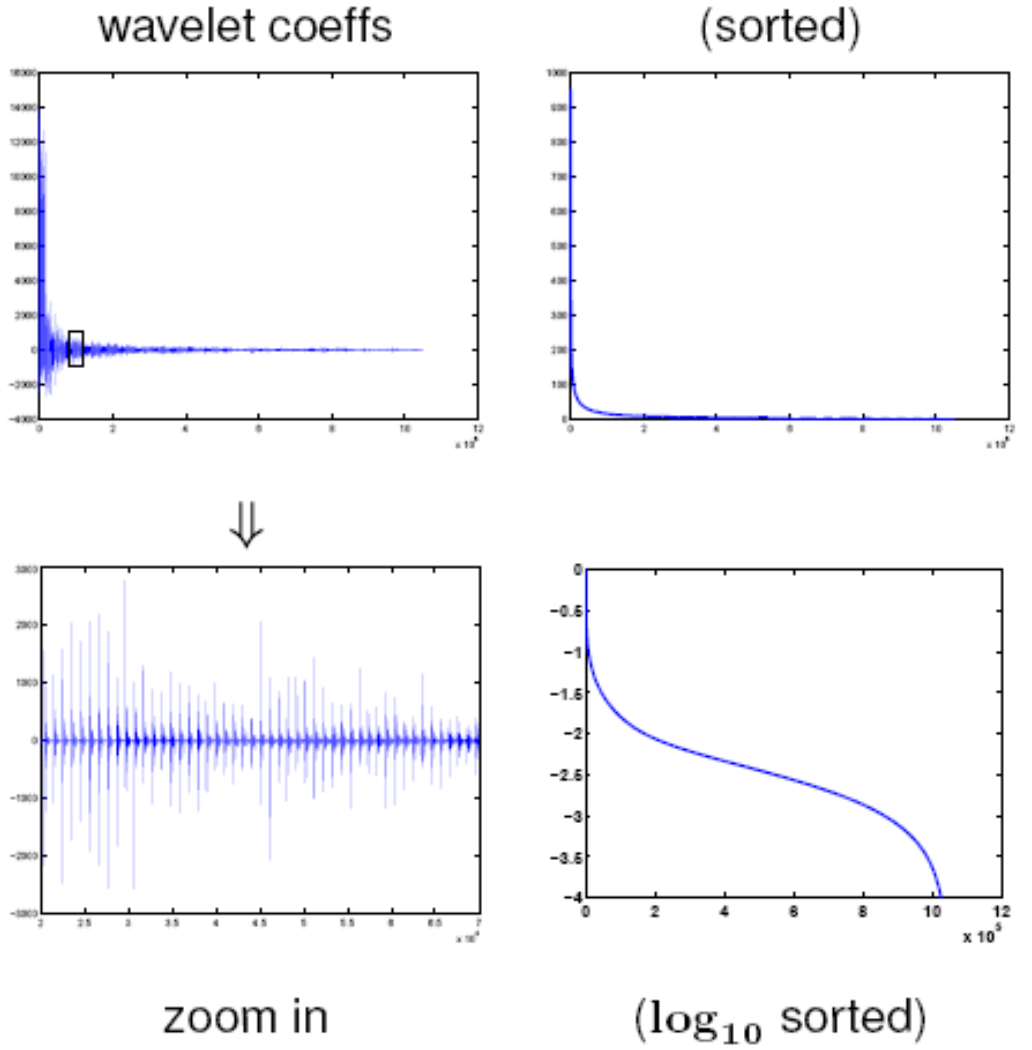
$$\sum_{i=\tilde{m}+1}^{m} \left(c_{\sigma(i)}\right)^2 \leq \varepsilon^2$$

Natural Signals are often sparse



Sparsity of the Signal

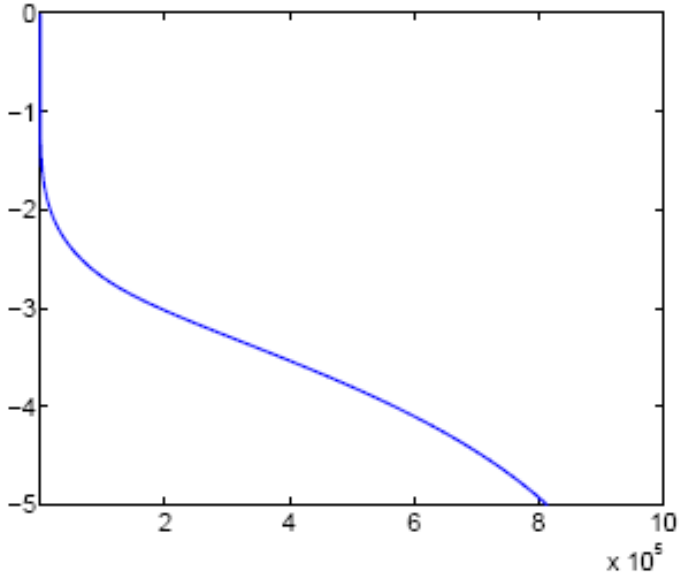
1 megapixel image



Decay of Coefficients

1 megapixel image

25k term approx



approx error

- Within 2 digits (in MSE) with $\approx 2.5\%$ of coeffs
- Original image = f, K-term approximation = f_K

$$\|f - f_K\|_2 \approx .01 \cdot \|f\|_2$$

Other Types of Wavelets

- Mexican Hat
- Daubechies Wavelets
- …..

- To better represent edges:
- Curvelets
- Ridgelets

Properties of Wavelets

- Multi-resolution
- Locality both in time/space domain and frequency domain
- (bi)Orthogonality
- Smooth basis functions
- Good approximations in L^2
- Fast transform algorithm O(n)
- Stable decomposition
- Sparse coefficients

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Edge-Avoiding À-Trous Wavelets

Combining Wavelets and Biltateral Filtering

Edge-Avoiding À-Trous Wavelet Transform for fast Global Illumination Filtering

[Dammertz, Sewtz, Hanika, Lensch – HPG 2010]

Edge-Avoiding À-Trous Wavelet Transform for fast Global Illumination Filtering

[Dammertz, Sewtz, Hanika, Lensch – HPG 2010]

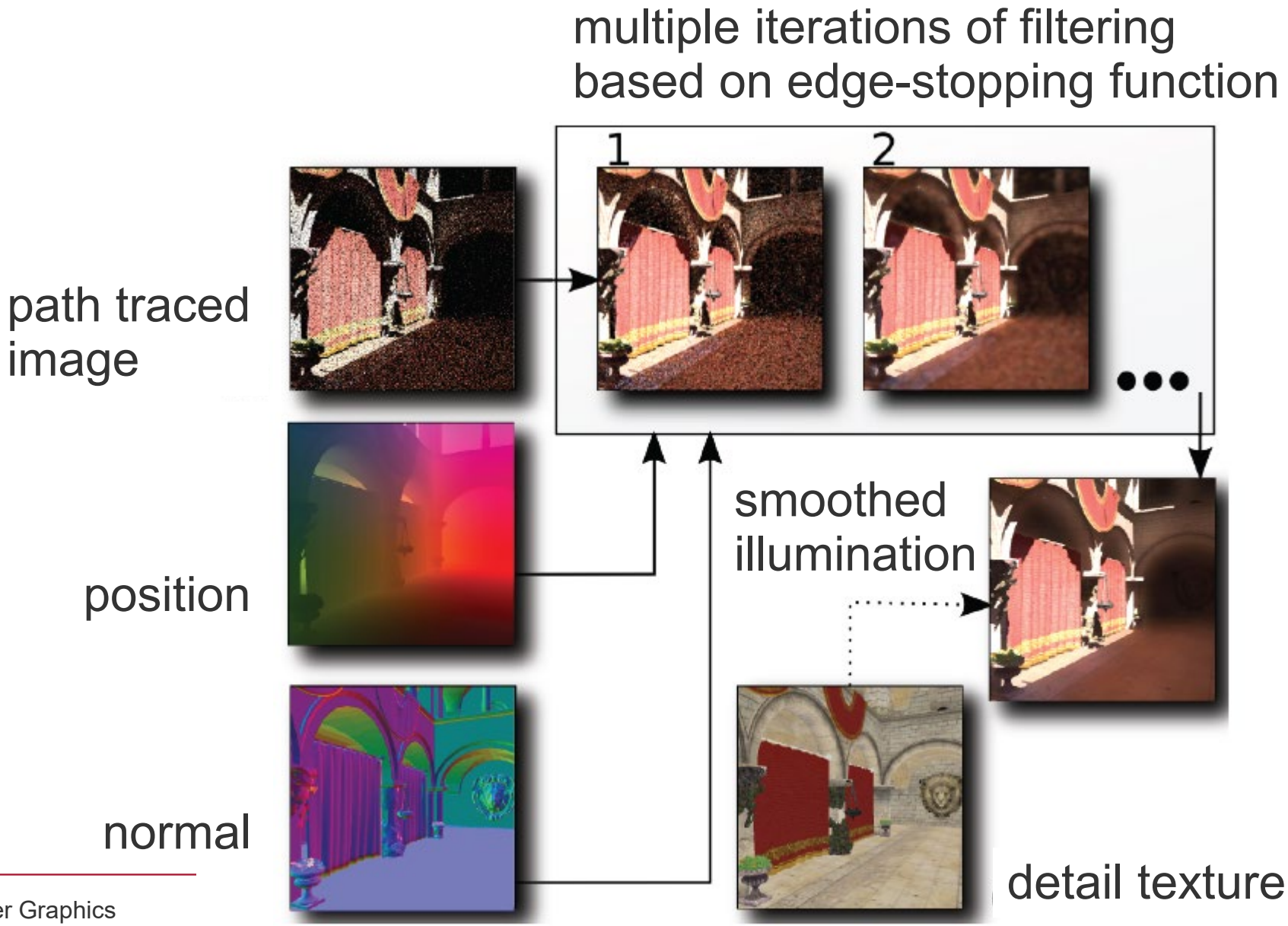
HPG 2010

paper1029

Edge-Avoiding A-Trous Wavelet Transform for fast Global Illumination Filtering

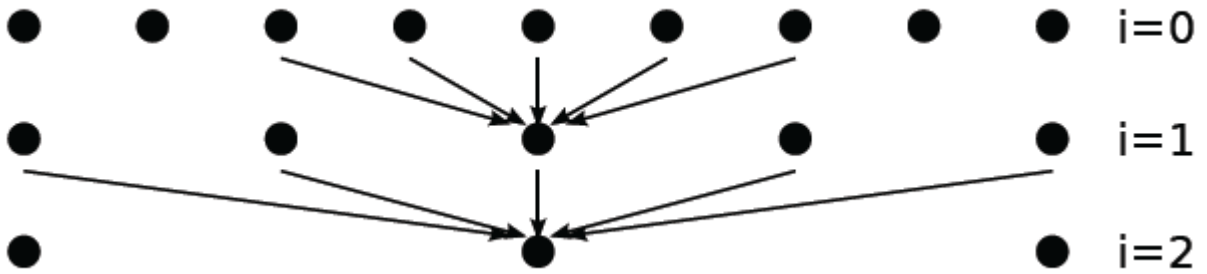
supplemental Material

Pipeline


multiple iterations of filtering
based on edge-stopping function
1
2
path traced
image
position
normal
smoothed
illumination
detail texture

Decimating Wavelet Filtering

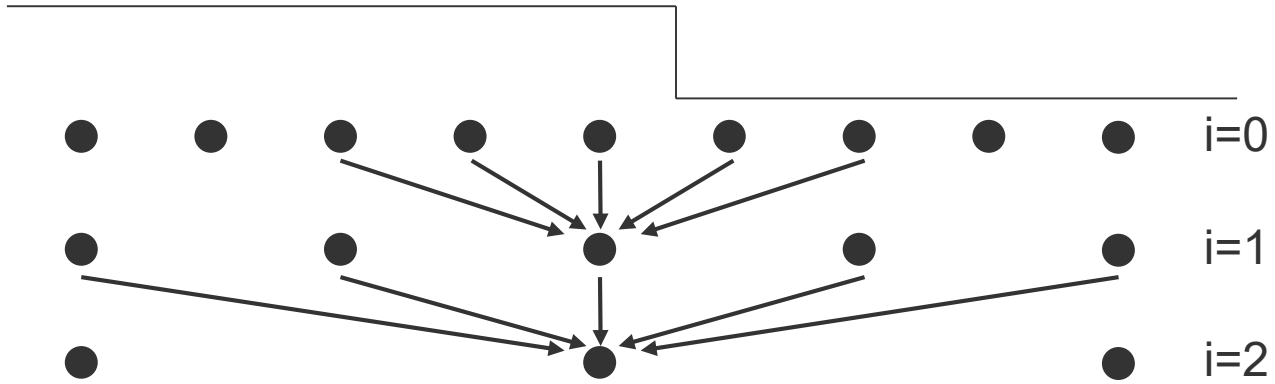
- Apply the same filter multiple times
  - reduce the image resolution at each iteration



- Benefits:
  - multi-scale representation
  - fast

Decimating Wavelet Transform

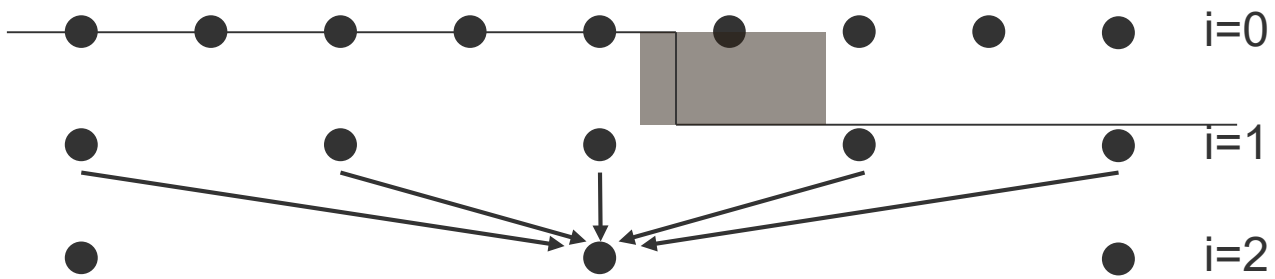
- Apply the same filter multiple times
  - reduce the image resolution at each iteration



- Benefits:
  - multi-scale representation, fast!
- Drawback: filtered information at few locations only
  - Where is the edge?

Decimating Wavelet Transform

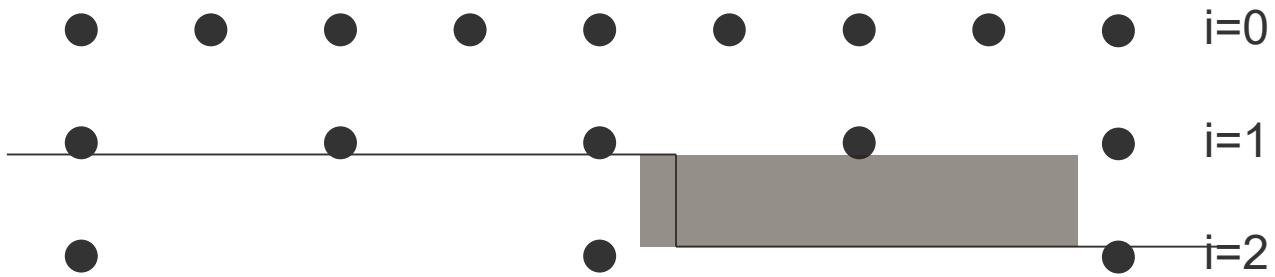
- Apply the same filter multiple times
  - reduce the image resolution at each iteration



- Benefits:
  - multi-scale representation, fast!
- Drawback: filtered information at few locations only
  - Where is the edge?

Decimating Wavelet Transform

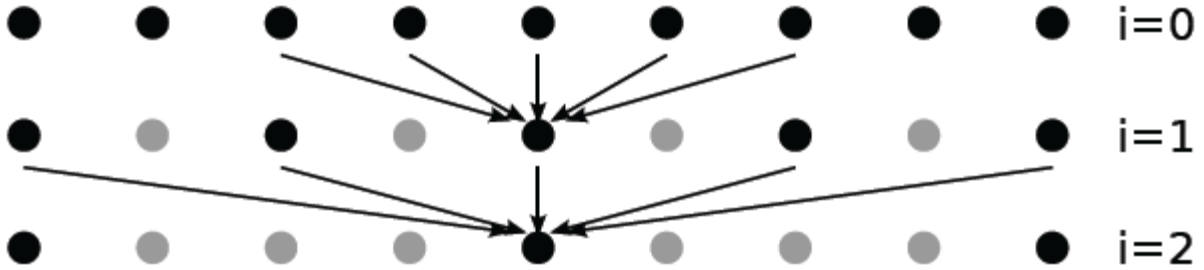
- Apply the same filter multiple times
  - reduce the image resolution at each iteration



- Benefits:
  - multi-scale representation, fast!
- Drawback: filtered information at few locations only
  - Where is the edge?

À-Trous Wavelet Filtering

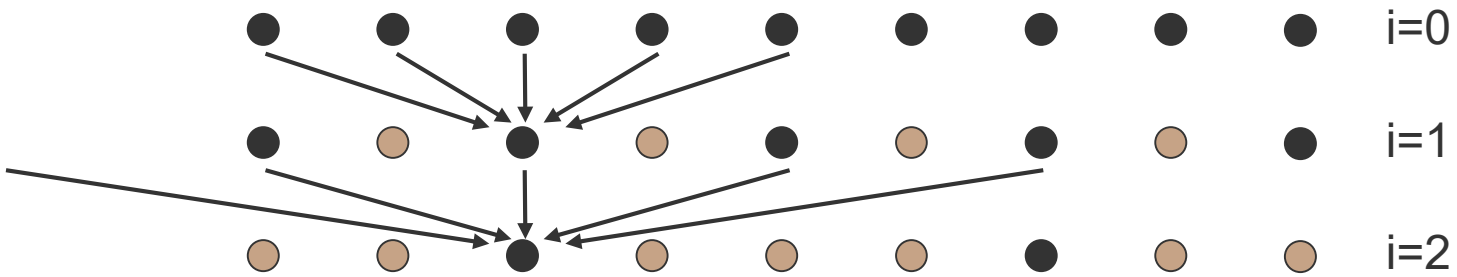
- “With holes”
- At each iteration convolve with a Gaussian
  - double the filter size
  - by introducing more and more holes



- Benefits:
  - constant effort per iteration (in contrast to undecimated wavelets)
  - filtered information at each pixel (in contrast to decimated wavelets)

À-Trous Wavelet Transform

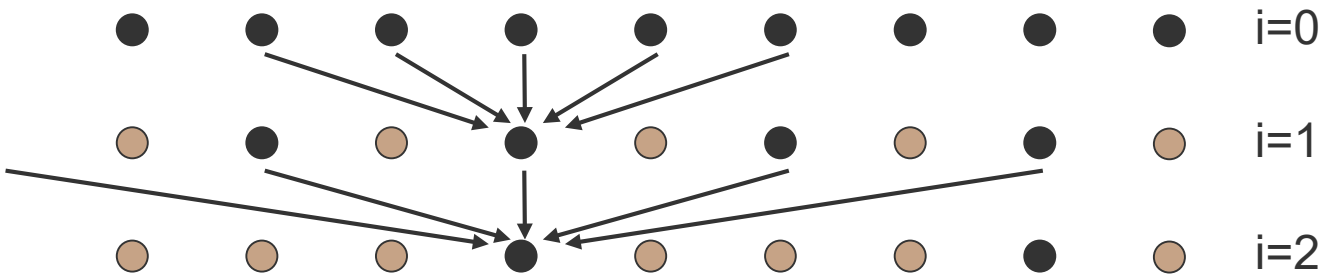
- “With holes”
- At each iteration convolve with a Gaussian
  - double the filter size
  - by introducing more and more holes



- Benefits:
  - constant effort per iteration (in contrast to undecimated wavelets)
  - filtered information at each pixel (in contrast to decimated wavelets)

À-Trous Wavelet Transform

- “With holes”
- At each iteration convolve with a Gaussian
  - double the filter size
  - by introducing more and more holes

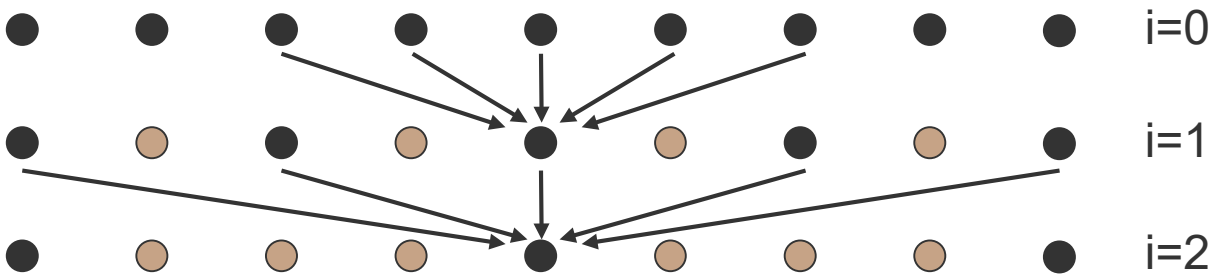


- Benefits:
  - constant effort per iteration (in contrast to undecimated wavelets)
  - filtered information at each pixel (in contrast to decimated wavelets)

À-Trous Wavelet Transform

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- “With holes”
- At each iteration convolve with a Gaussian
  - double the filter size
  - by introducing more and more holes



- Benefits:
  - constant effort per iteration (in contrast to undecimated wavelets)
  - filtered information at each pixel (in contrast to decimated wavelets)

À-Trous Wavelet Transform

1. At level $i = 0$, start with input signal $c_0(p)$.
2. $c_{i+1}(p) = c_i(p) * h_i$, where $*$ is the discrete convolution.

   The distance between the entries in the filter h_i is 2^i.
3. $d_i(p) = c_i(p) - c_{i+1}(p)$

   where d_i are the detail or wavelet coefficients of level
4. Repeat 2 and 3 until $i = N$ (number of levels to compute)
5. $\{d_0, d_1, \ldots, d_{N-1}, c_N\}$ is the wavelet transform of c .
6. The reconstruction is given by $c = c_N + \sum_{i=0}^{N-1} d_i$

À-Trous Wavelet *Filtering*

1. At level $i = 0$, start with input signal $c_0(p)$.
2. $c_{i+1}(p) = c_i(p) * h_i$, where $*$ is the discrete convolution.

   The distance between the entries in the filter h_i is 2^i.
3. $d_i(p) = c_i(p) - c_{i+1}(p)$

   where d_i are the detail or wavelet coefficients of level
4. Repeat 2 and 3 until $i = N$ (number of levels to compute)
5. $\{d_0, d_1, \ldots, d_{N-1}, c_N\}$ is the wavelet transform of c .
6. The **filtered** reconstruction is given by $c = c_N + \sum_{i=0}^{N-1} \alpha_i d_i$

Edge-Stopping Function

- Compute weighted convolution
  (compare to bilateral filter)

$$c_{i+1}(p) = \frac{\sum_{q \in Q} h_i(q) \cdot w(p,q) \cdot c_i(p)}{\sum_{q \in Q} h_i(q) \cdot w(p,q)}$$

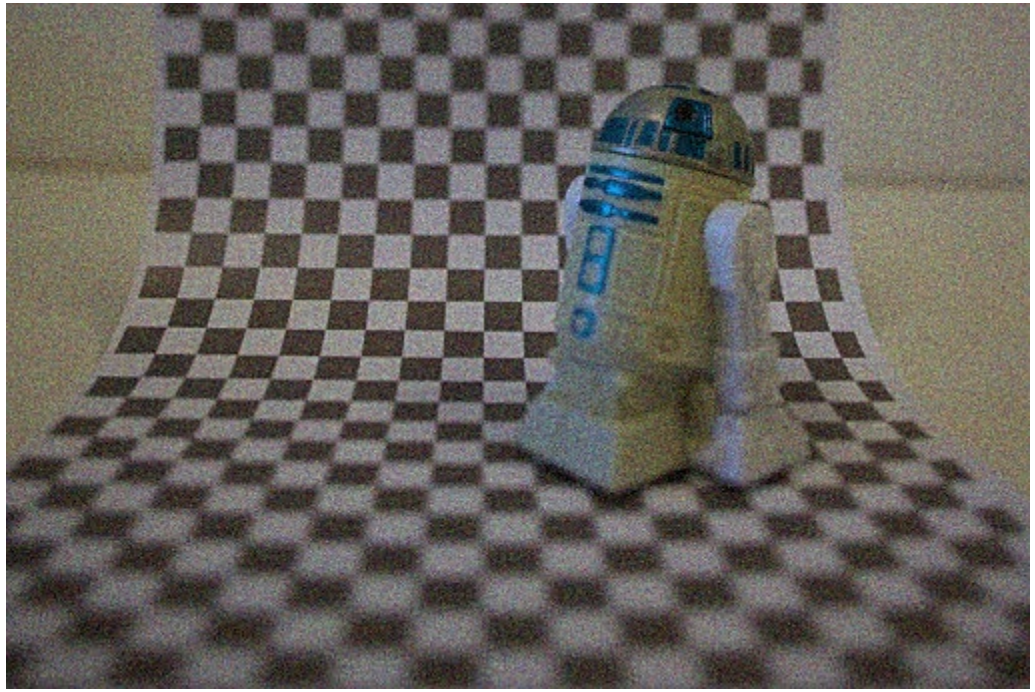
$$\left[c_{i+1}(p) = c_i(p) * h_i \right]$$

- With weights $w(p,q) = w_X(p,q)$ with

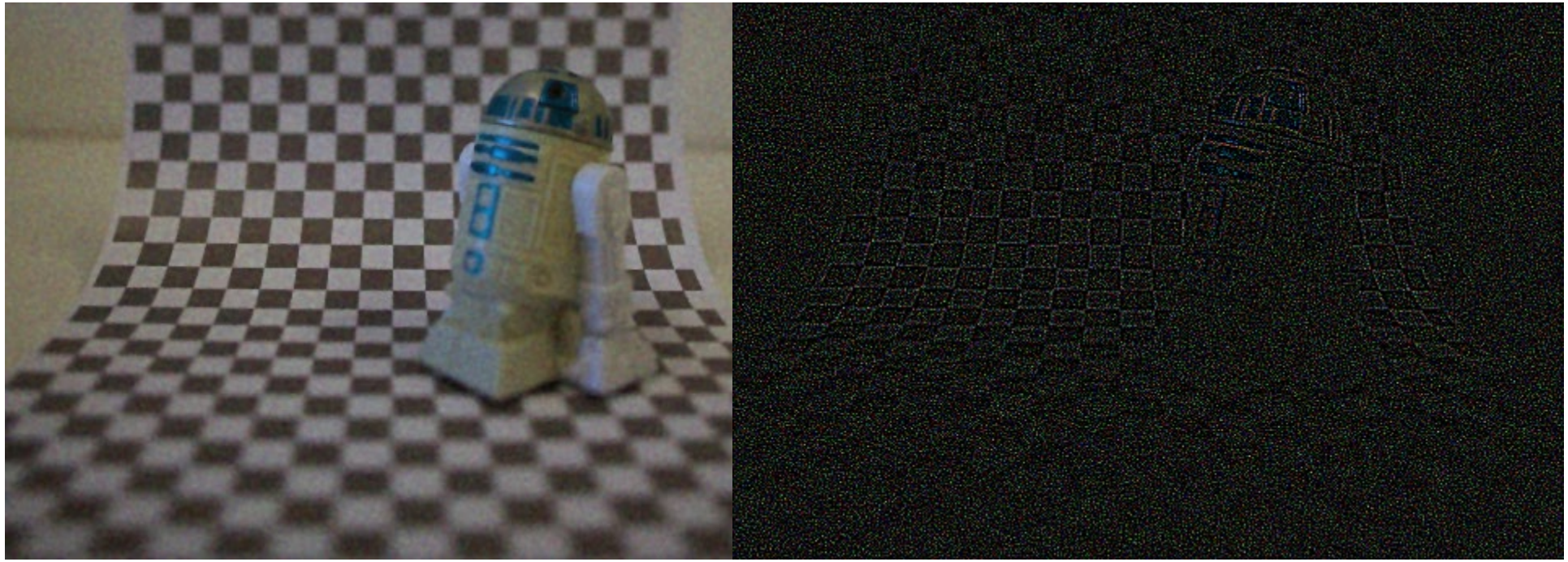
$$w_X(p,q) = e^{\left(-\frac{\|I_p - I_q\|^2}{{\sigma_X}^2} \right)}$$

- σ_X can be made large for the first two levels

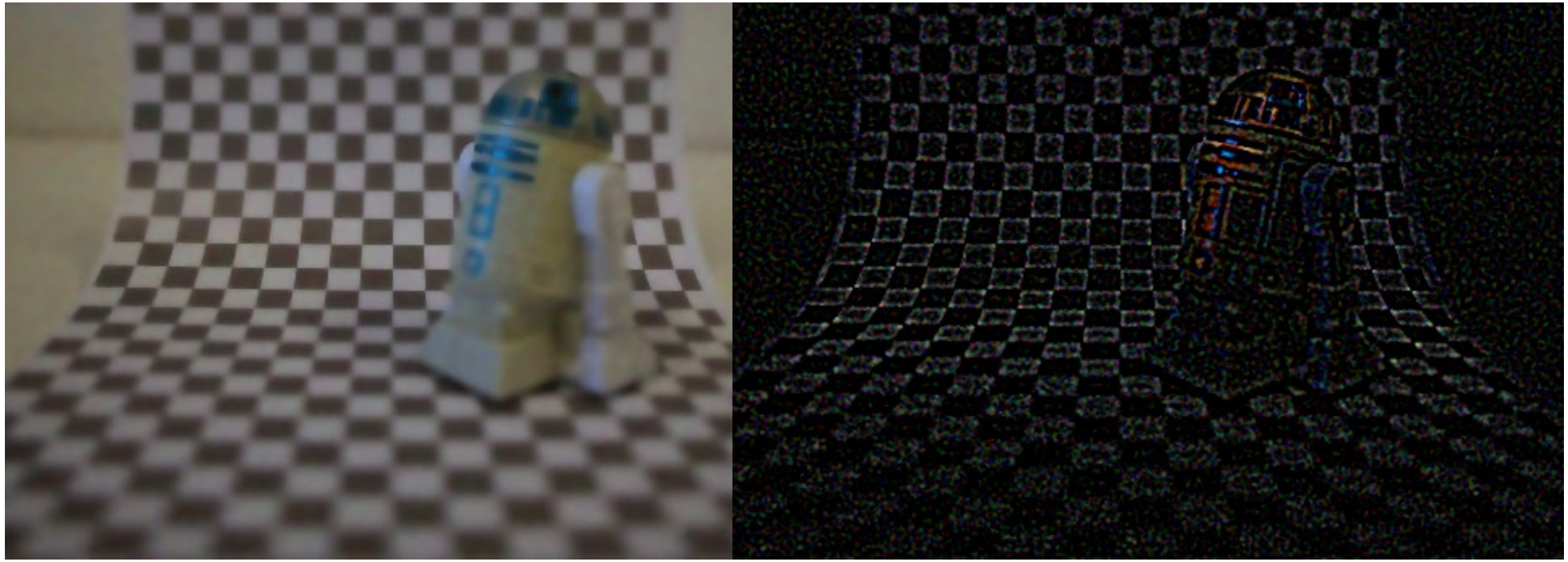
Input



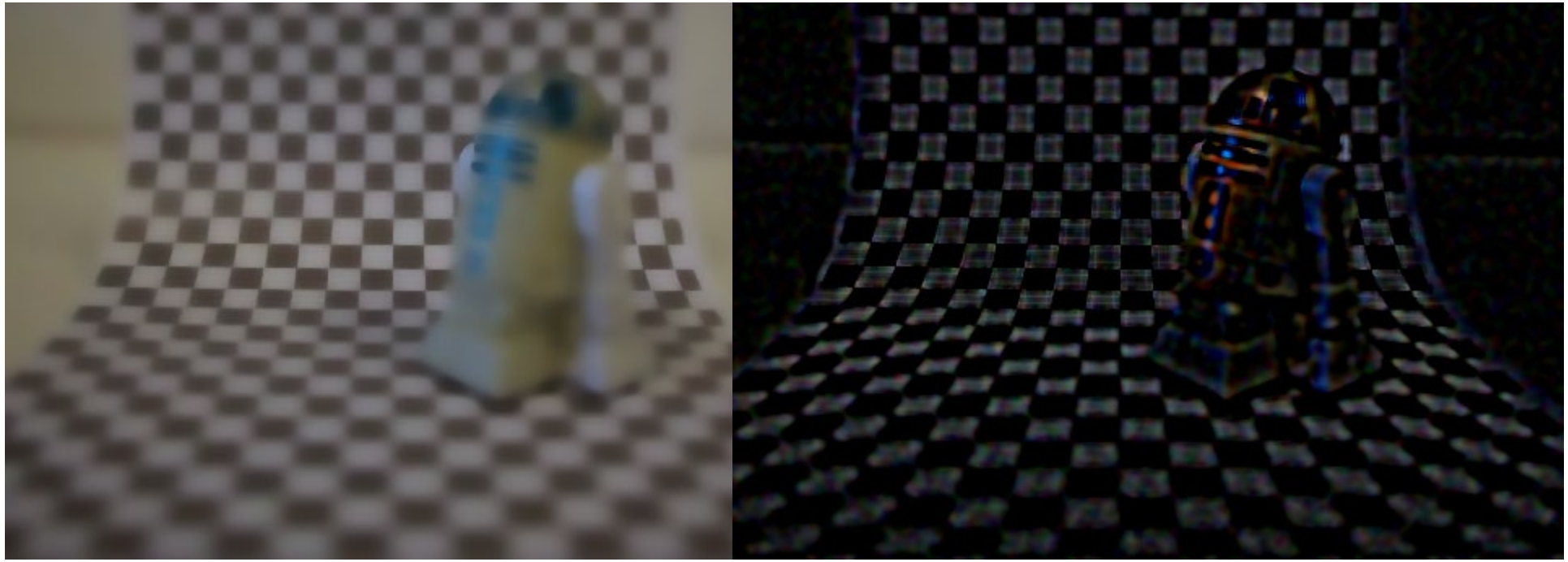
Level 0



Level 1



Level 2



Denoising by Shrinkage

- Estimate standard deviation of the noise in the image:

$$\sigma_n = \frac{median(|d_0|)}{0.6745}.$$

- Compute optimal shrinkage threshold based on the signal variance

$$T = \frac{\sigma_{n,i}^2}{\sqrt{\max\{0, \sigma_{y,i}^2 - \sigma_{n,i}^2\}}}$$

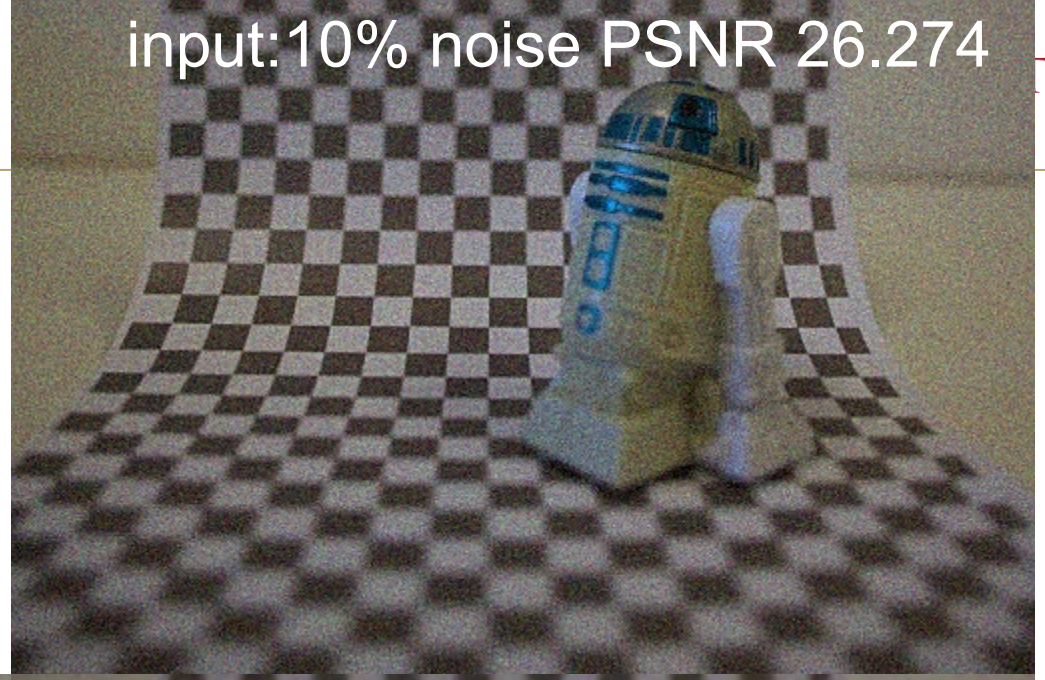
$$\text{with } \sigma_{y,i}^2 = \frac{1}{N}\sum_p d_i(p)^2 \text{ and } \sigma_{n,i} = \sigma_n \cdot 2^{-i}.$$

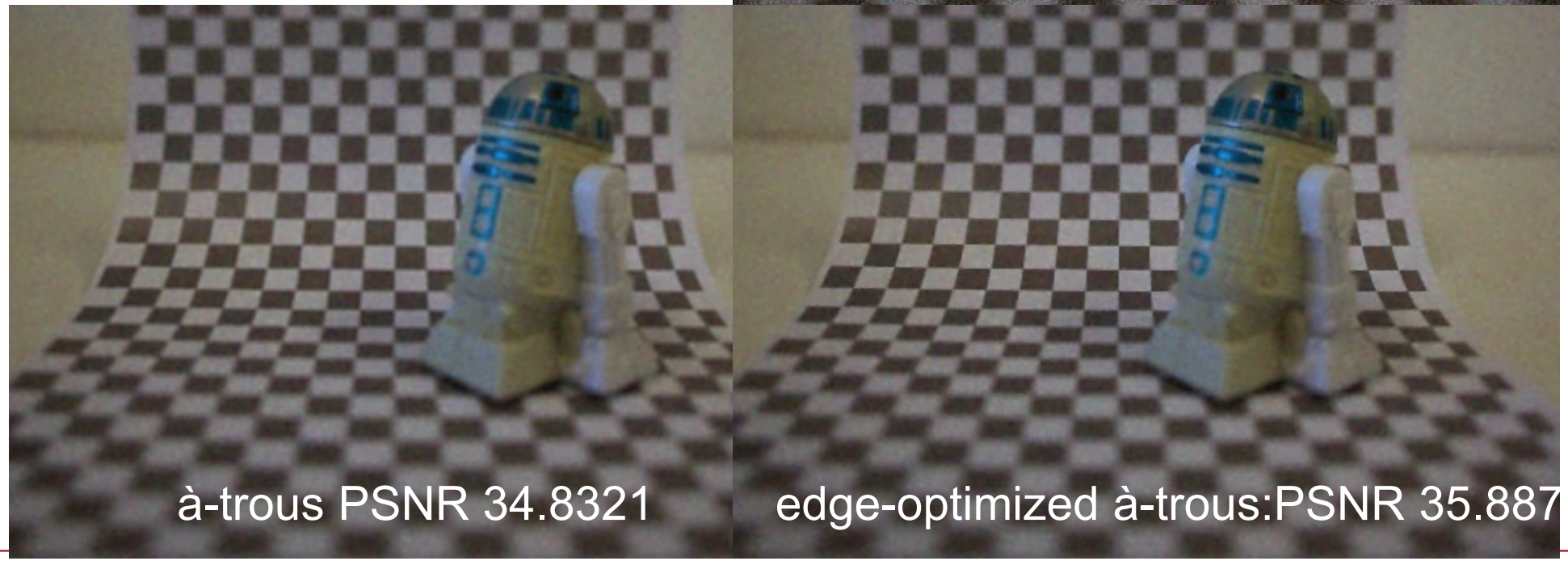
- Apply shrinkage and contrast boost (if wanted):

$$d_i' = \max\{0, |d_i| - T\} \cdot sign(d_i)$$

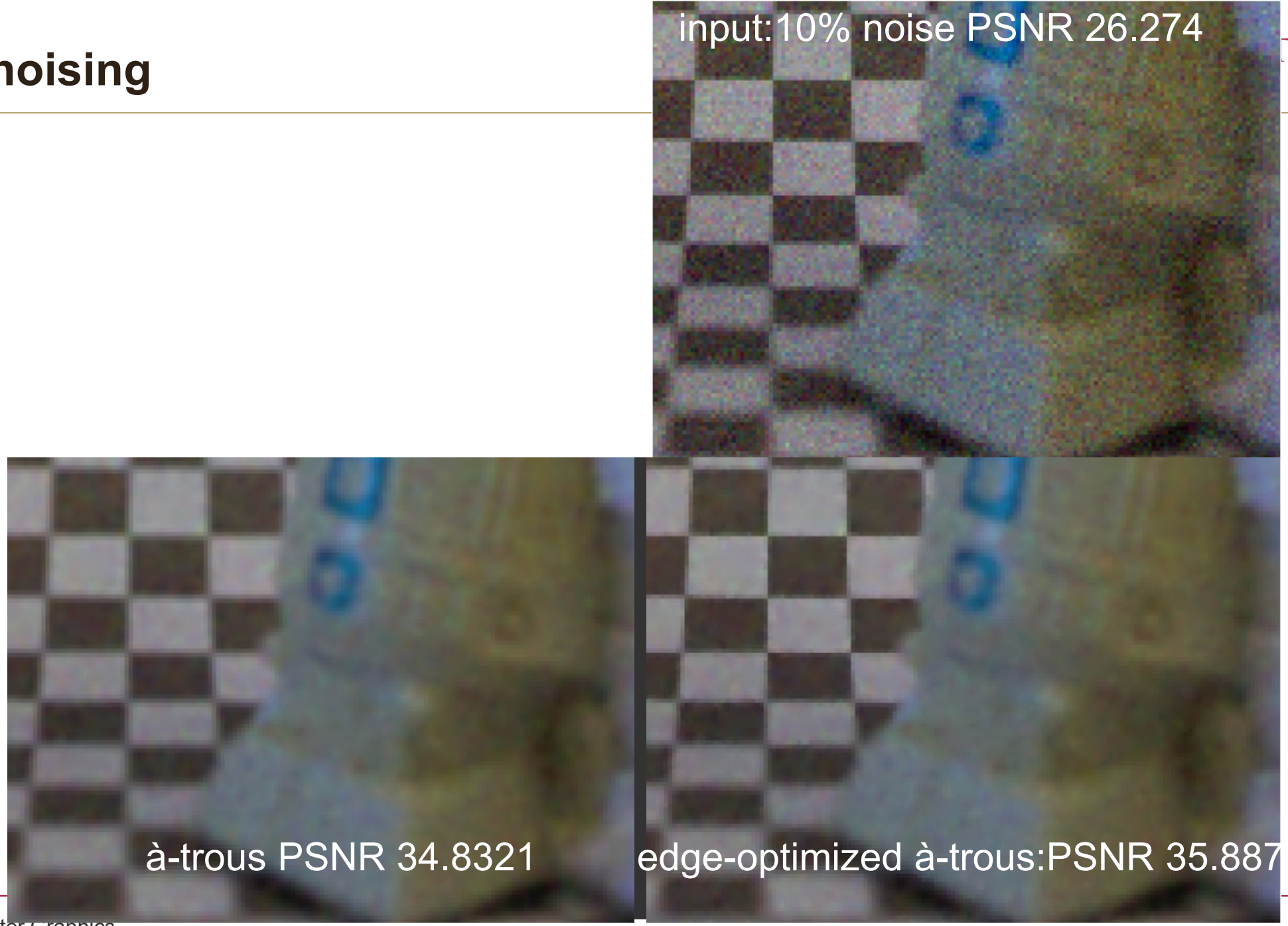
$$\text{and } c_{i-1} = c_i + \beta \cdot d_i',$$

Denoising

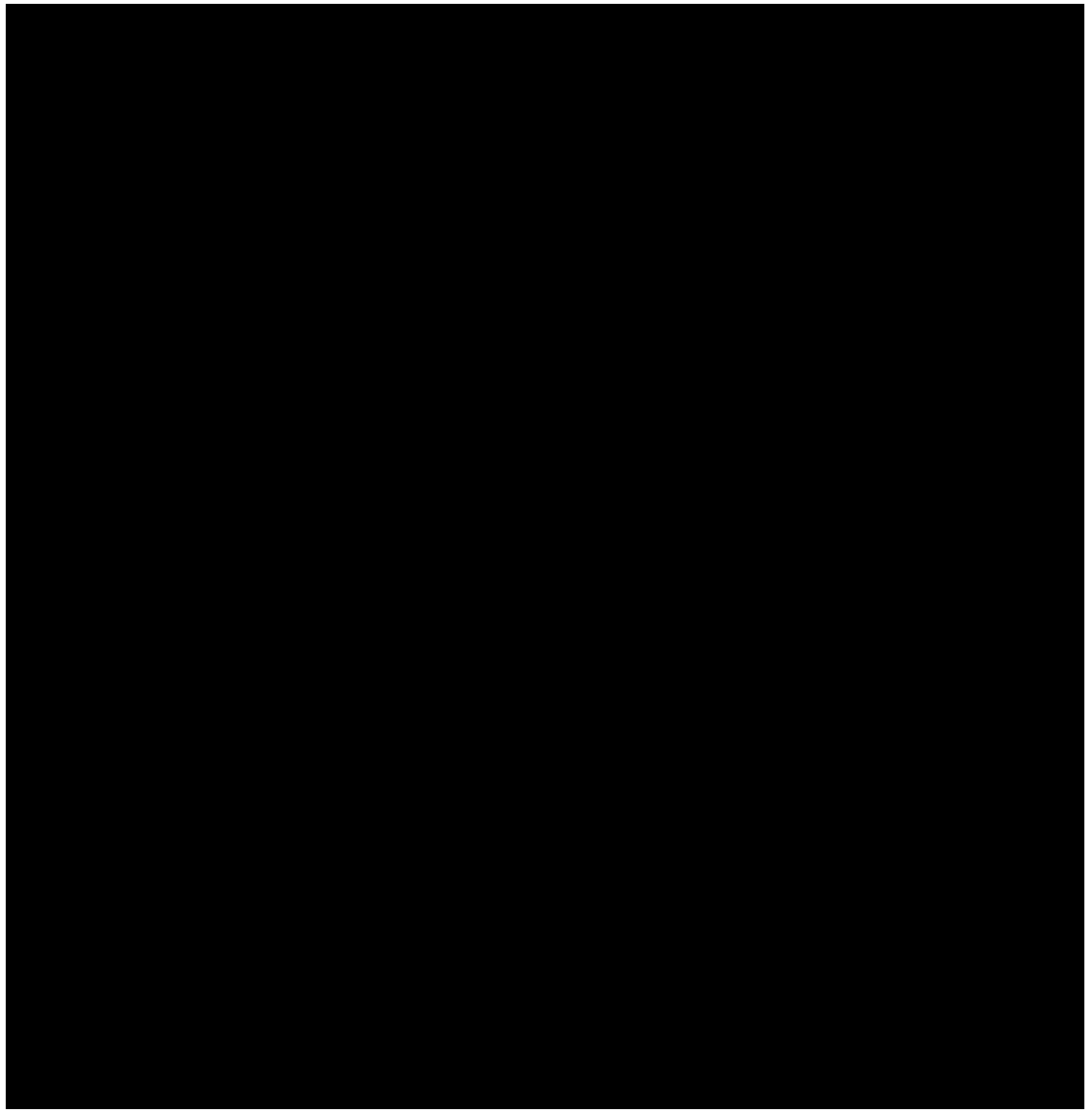

input:10% noise PSNR 26.274


à-trous PSNR 34.8321
edge-optimized à-trous:PSNR 35.887

Denoising


input:10% noise PSNR 26.274
à-trous PSNR 34.8321
edge-optimized à-trous:PSNR 35.887

Contrast Enhancement



Edge-Avoiding À-Trous Wavelets – Contrast Enhancement

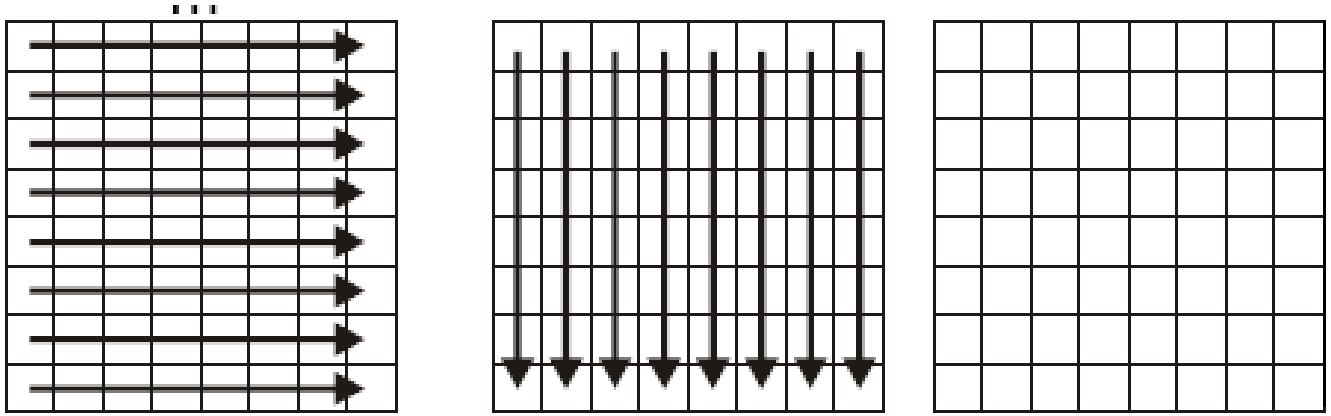
Edge-Avoiding À-Trous Wavelets – Smoothing

Wrap-Up

Gaussian - Implementation

- Gaussian filter is separable

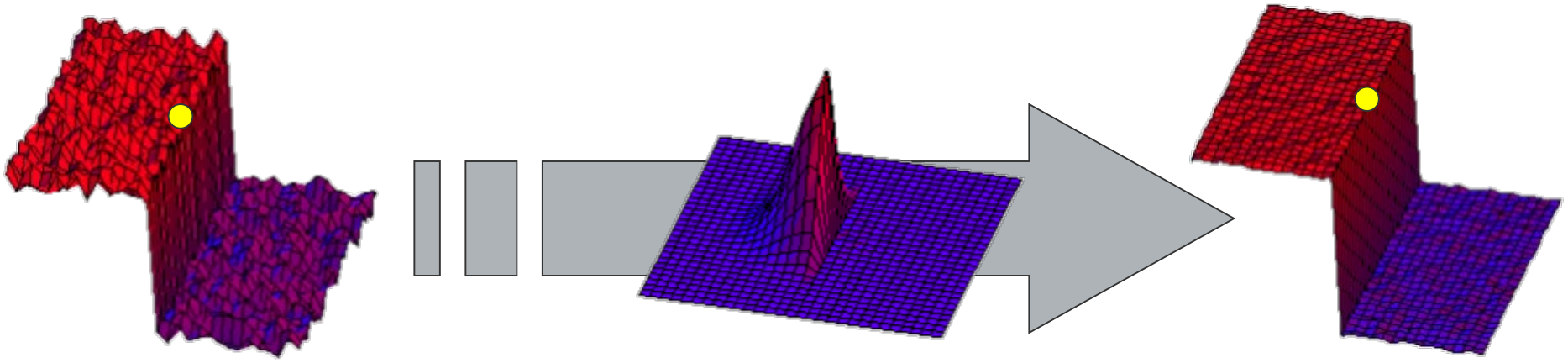
$$\bar{f}(x,y) = G_{(x,y)} \otimes f(x,y) = G_y \otimes (G_x \otimes f(x,y))$$

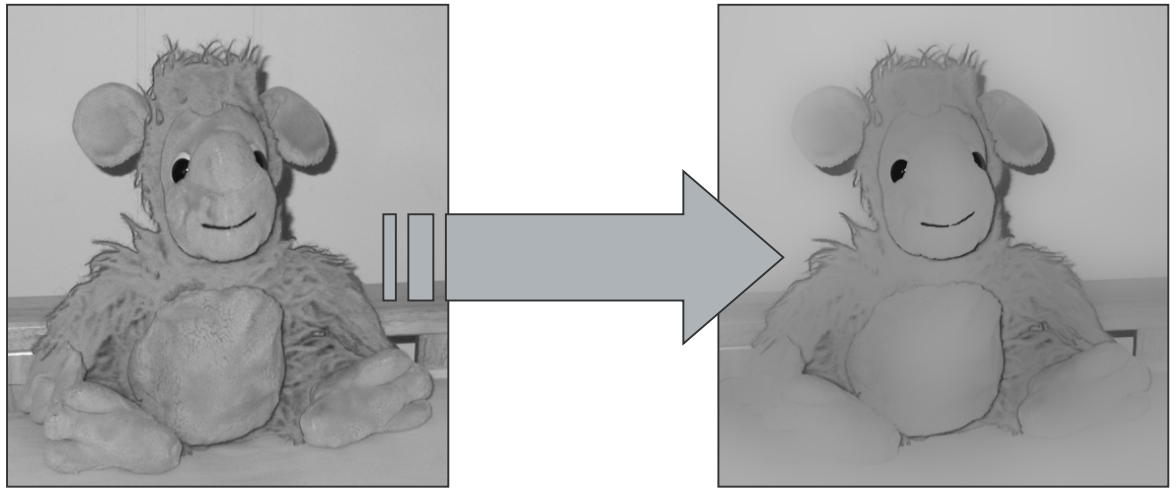


$G_x \otimes f(x,y)$ $G_y \otimes G_x \otimes f(x,y)$ $\bar{f}(x,y)$

Large-scale Layer

- Bilateral filter – edge preserving filter
- Smith and Brady 1997; Tomasi and Manducci 1998; Durand et al. 2002

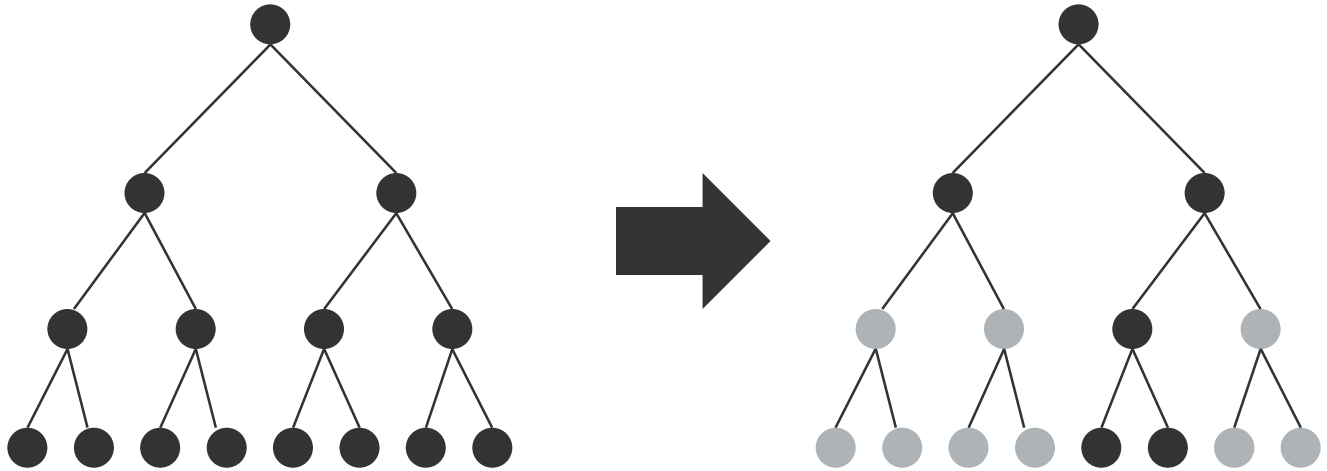




Edge-Avoiding À-Trous Wavelets – Smoothing

Sparse Representation

- First measure entire matrix
- Then compress by determining the sparse tree
  - since all information is given you are on the save side



Summary

- Simple Filters
  - Gauss
  - Sobel
  - Median, …
- Bilateral
  - Upsampling
  - Cross-modal Upsampling
- Wavelets
  - Haar
  - À-Trous
  - Edge-avoiding À-Trous
  - Denoising