04.09.2020

Digital Image Processing (CSE/ECE 478)

Lecture-8: Bilateral Filtering, Linearity Intro to Frequency Domain Processing

Ravi Kiran

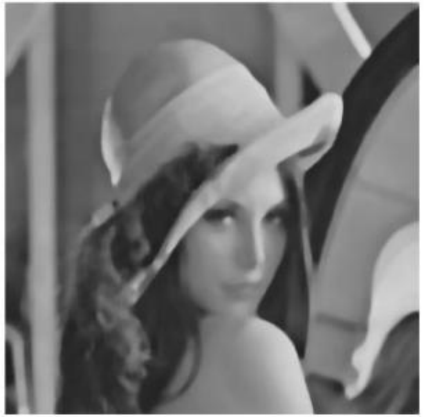
Center for Visual Information Technology (CVIT), IIIT Hyderabad

- Mean: blurs image, removes simple noise, no details are preserved
- Gaussian: blurs image, preserves details only for small σ.
- Median: preserves some details, good at removing strong noise

original

3x3 mean

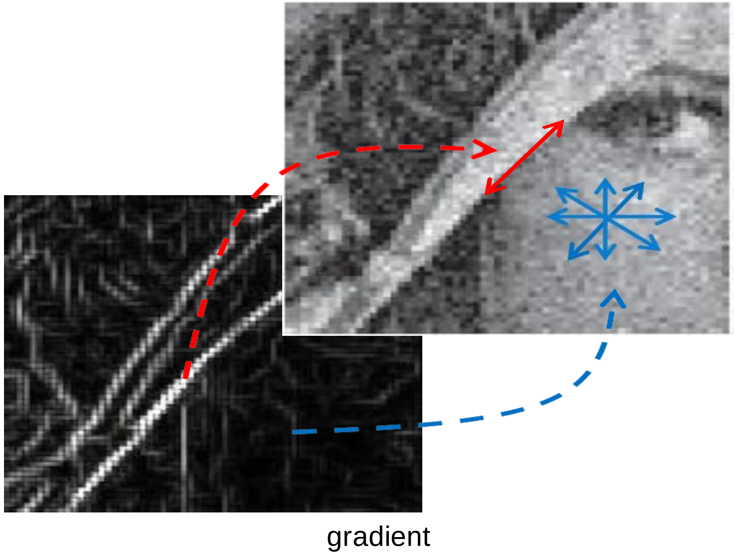
3x3 gaussian



3x3 median

Edge Preserving Filtering

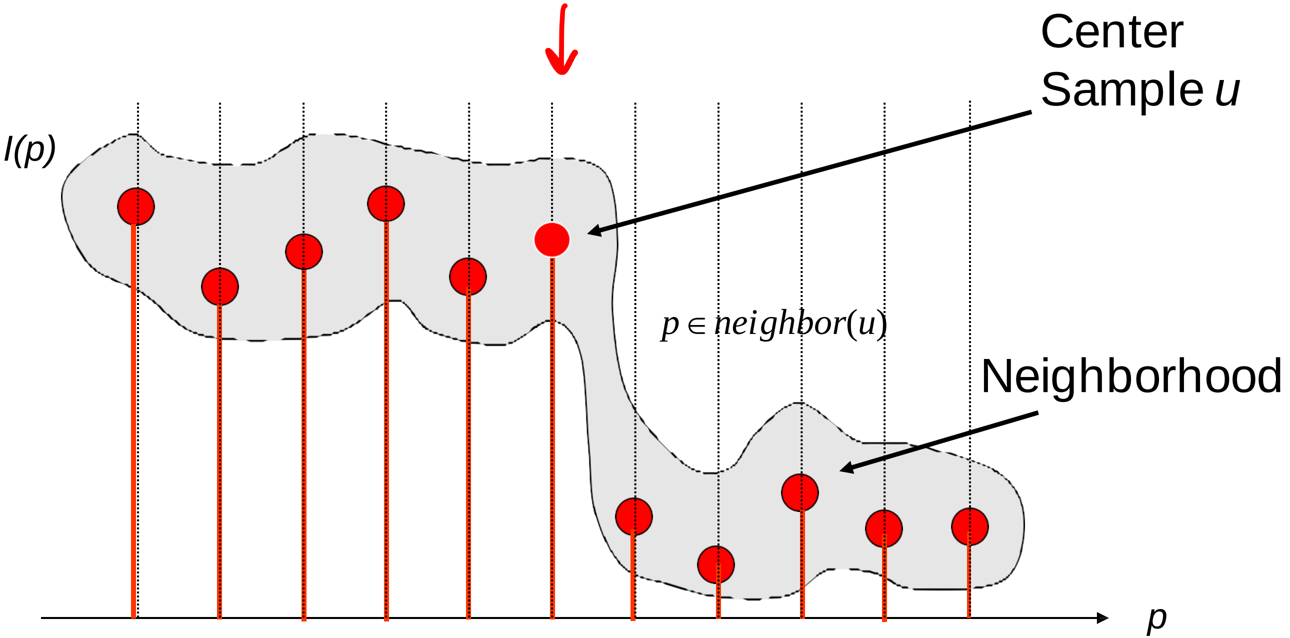
- Edges ⇒ smooth only along edges
- “Smooth” regions ⇒ smooth isotropically



Bilateral Filters - 1D example

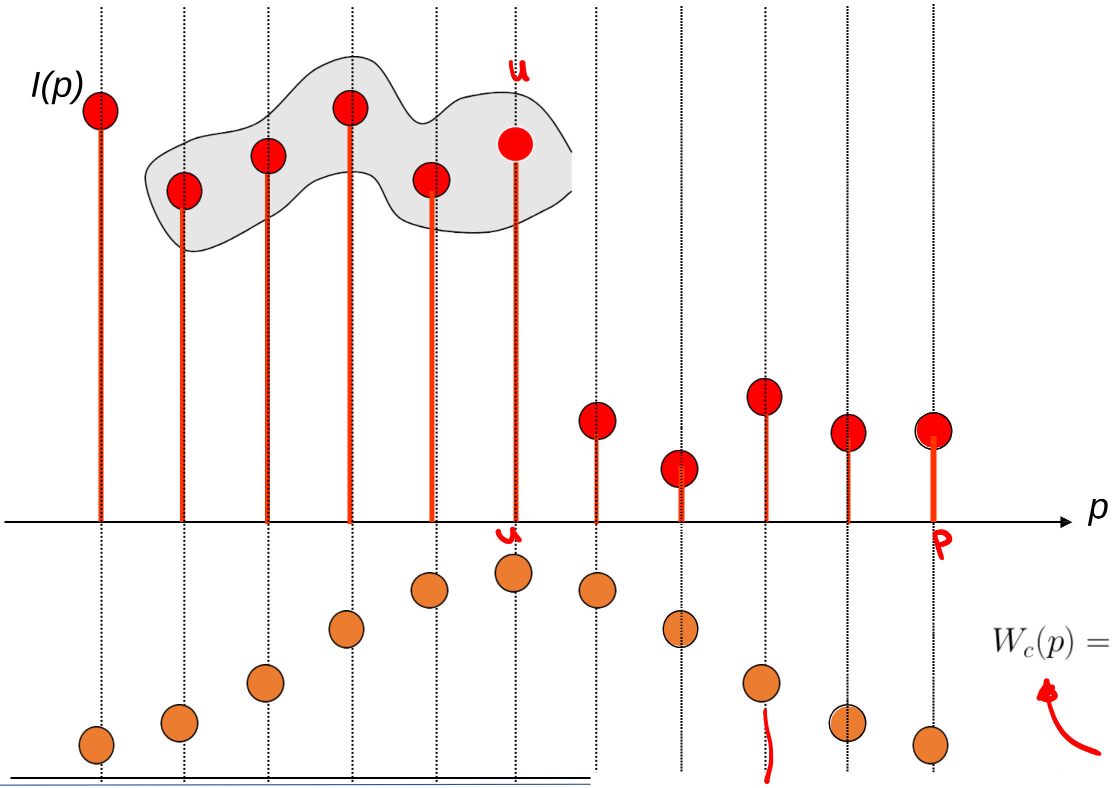
1×3

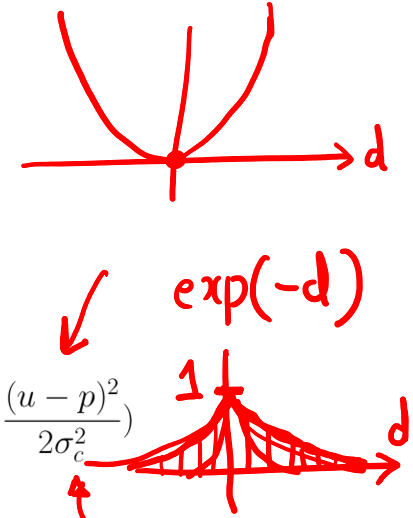
11×11



It is clear that in weighting this neighborhood,
we would like to preserve the step

Gaussian Weights

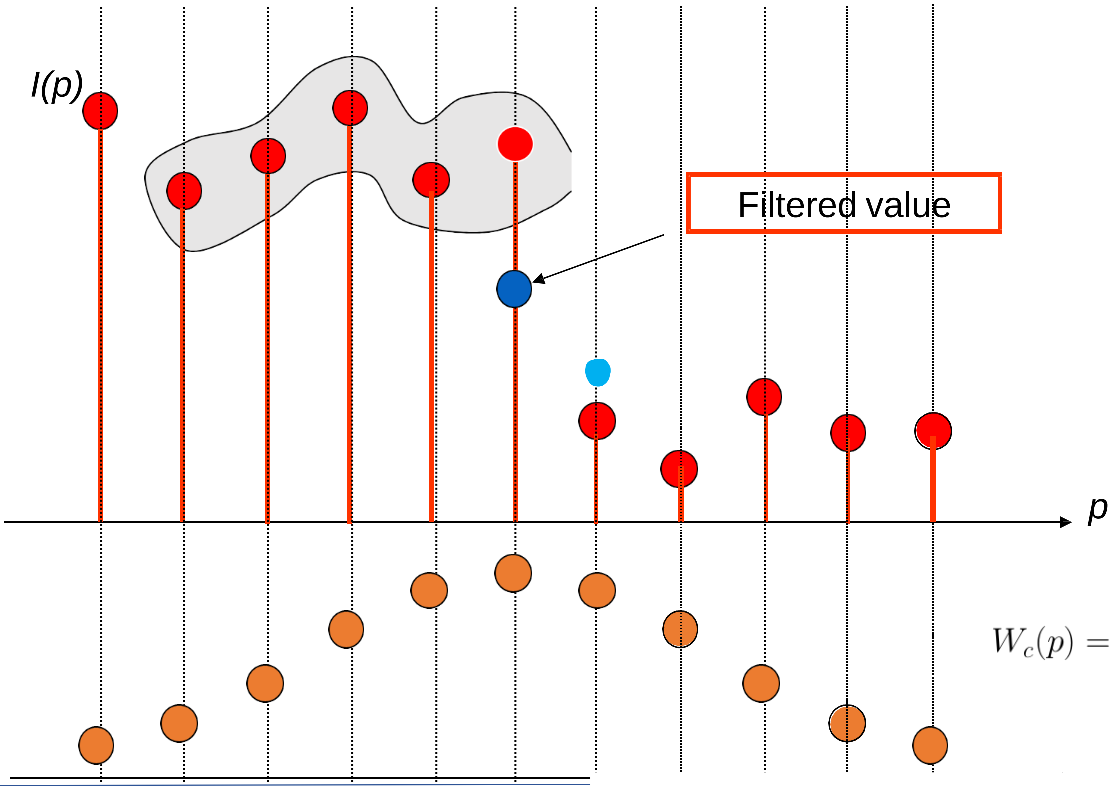




$$W_c(p) = exp(-\frac{(u-p)^2}{2\sigma_c^2})$$

Filtered Output

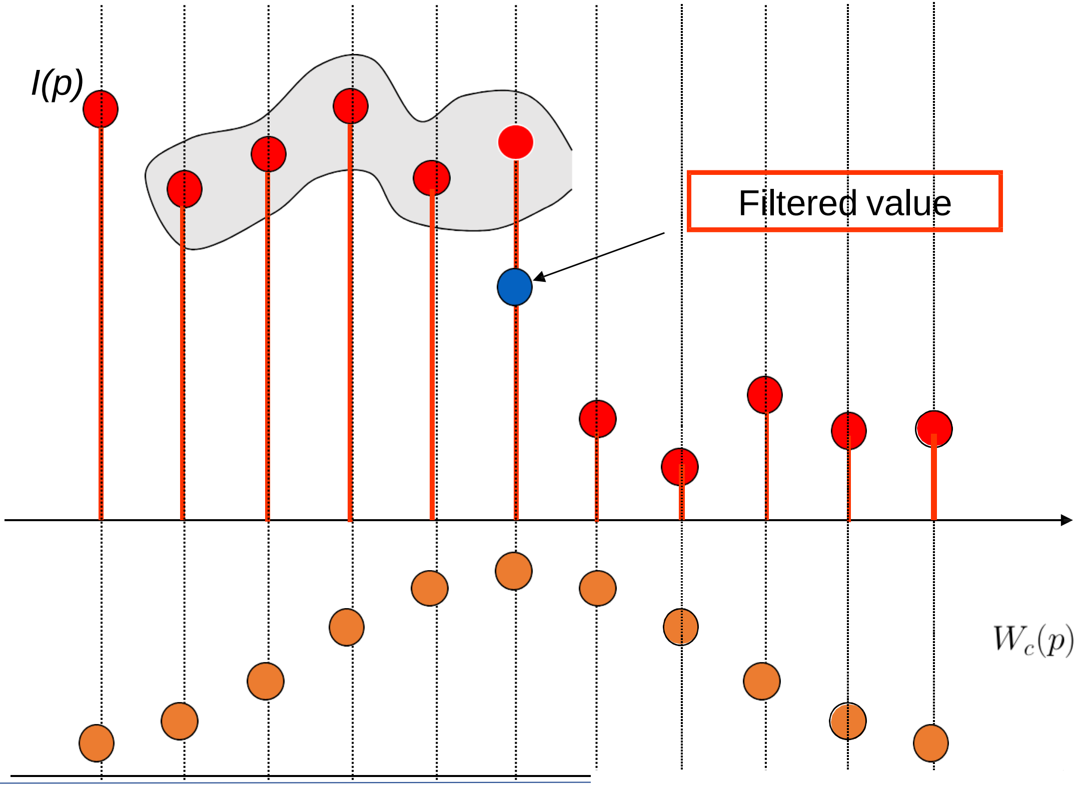
- Weighted sum on the $W_c(p)$



$$W_c(p) = exp(-\frac{(u-p)^2}{2\sigma_c^2})$$

Edge loss

- Edge is smoothed/lost

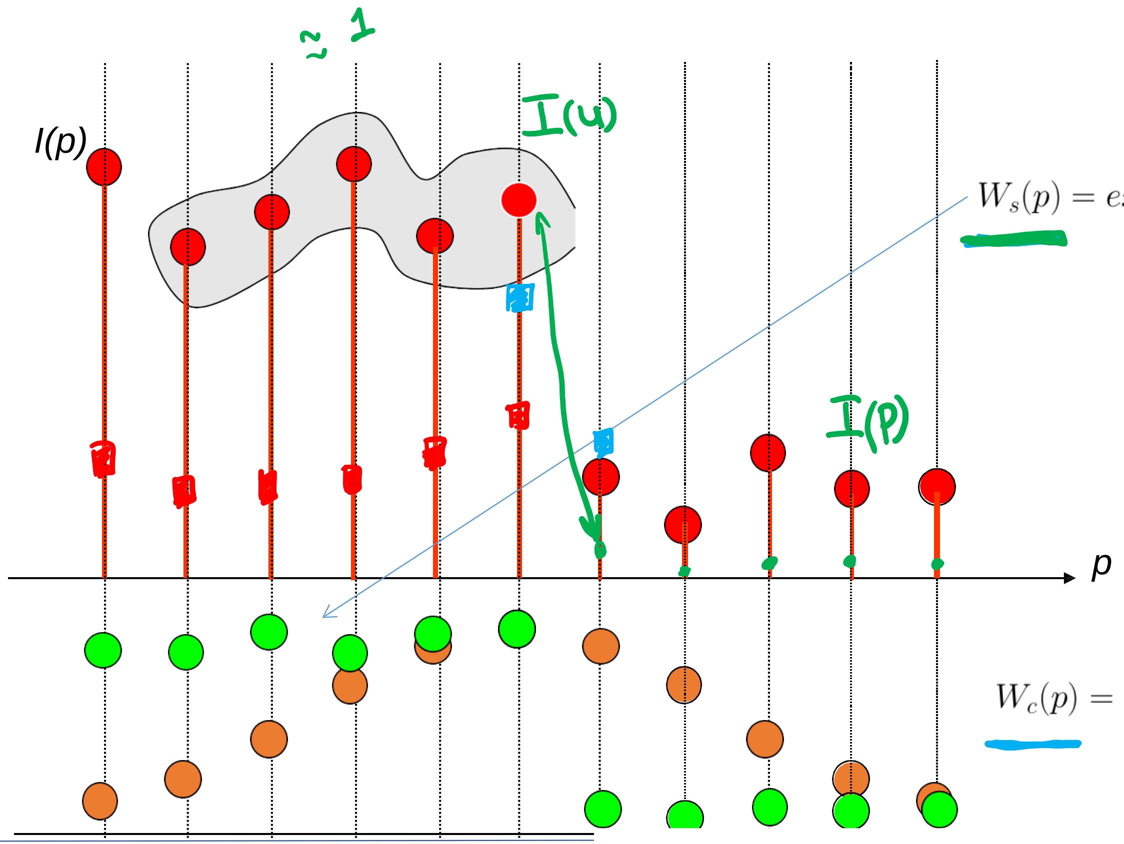


Geometric weights.

$$W_c(p) = exp(-\frac{(u-p)^2}{2\sigma_c^2})$$

Photometric Weights

- Introducing Photometric weights



$$W_s(p) = exp(-\frac{(I(u) - I(p))^2}{2\sigma_s^2})$$

$$W_{bi}(p) = W_s(p) W_c(p)$$

$$W_c(p) = exp(-\frac{(u - p)^2}{2\sigma_c^2})$$

Bilateral Filtering

- Fitler Weights derived from both geometric and photometric distances

Denoise

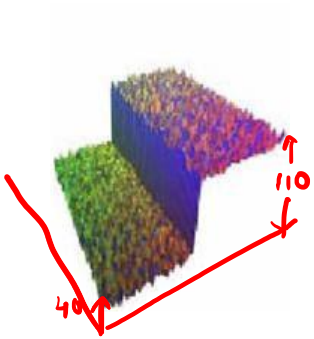
Feature preserving

$$I'(u) = \frac{\sum_{p \in N(u)} \underbrace{e^{-\frac{\|u-p\|^2}{2\sigma_c^2}}} \underbrace{e^{-\frac{|I(u)-I(p)|}{2\sigma_s^2}}} I(p)}{\sum_{p \in N(u)} e^{-\frac{\|u-p\|^2}{2\sigma_c^2}} e^{-\frac{|I(u)-I(p)|}{2\sigma_s^2}}}$$

Normalization

Bilateral Filter

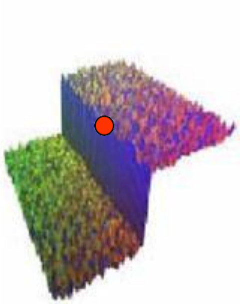
- Illustration of bilateral filter changes



input

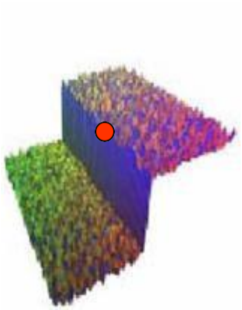
Bilateral Filter

- Illustration of bilateral filter changes

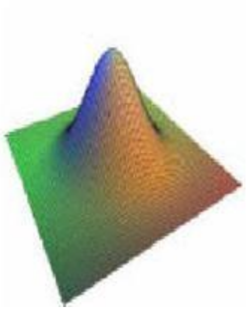


input

Bilateral Filter

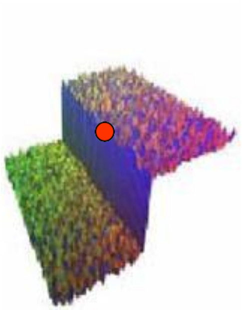


input

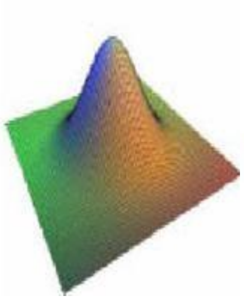


W_c

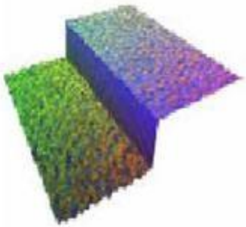
Bilateral Filter



input

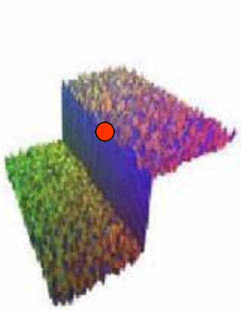


W_c

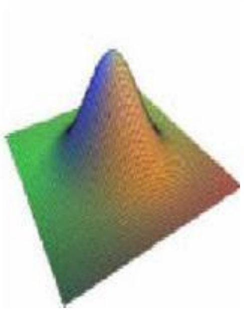


W_s

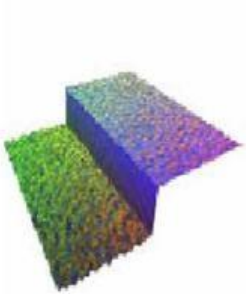
Bilateral Filter



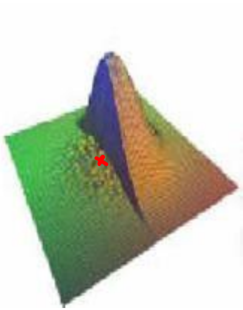
input



W_c



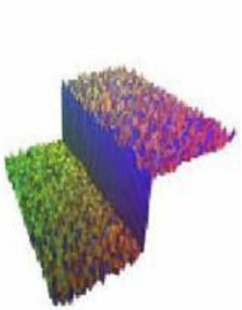
W_s



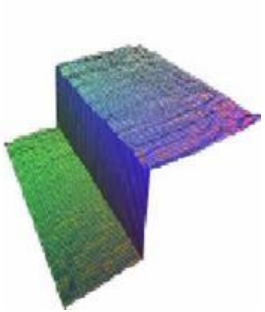
$W_s * W_c$

Bilateral Filter

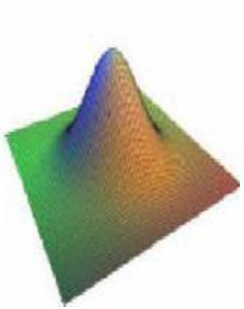
- Filtering process



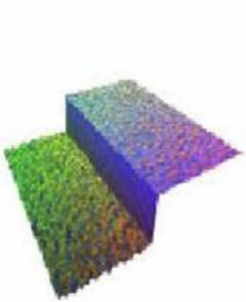
input



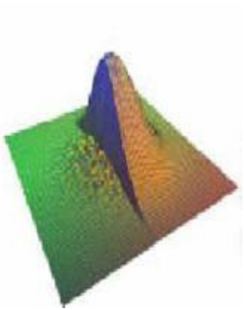
Output



W_c



W_s



W_s*W_c

Bilateral Filter Results

Original

Bilateral Filter Results

$\sigma_c = 3,$
$\sigma_s = 3$

Bilateral Filter Results

$\sigma_c = 6,$
$\sigma_s = 3$

Bilateral Filter Results

$\sigma_c = 12,$
$\sigma_s = 3$

Bilateral Filter Results

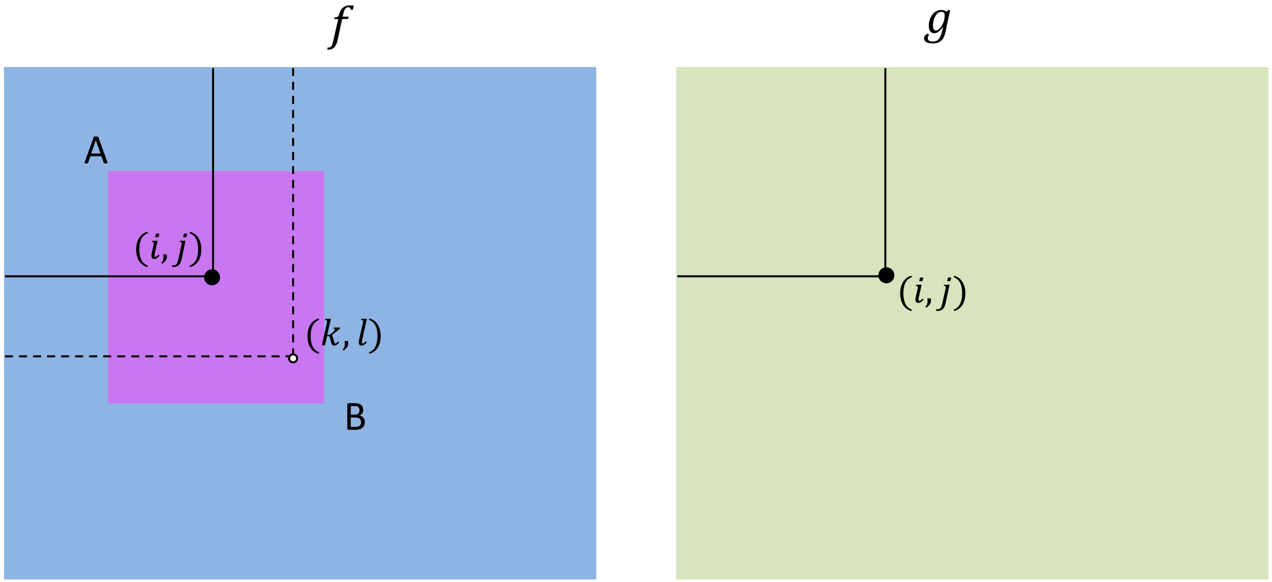
$$\sigma_c = 12, \quad \sigma_s = 6$$

Bilateral Filter Results

$$\sigma_c = 15,$$
$$\sigma_s = 8$$

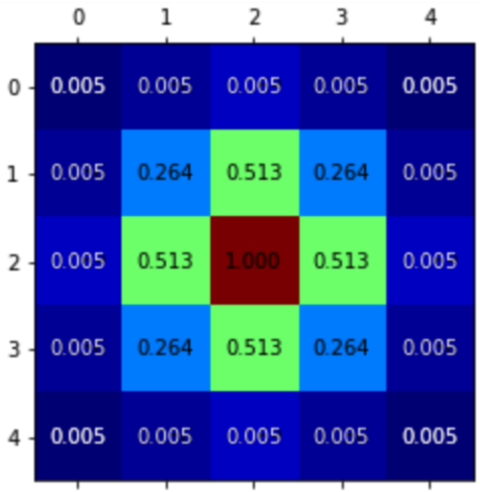
Linear Spatial Filter

$$I'(u,v) \leftarrow \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j) \cdot H(i,j)$$



$\sigma_d = 1.5$

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right)$$

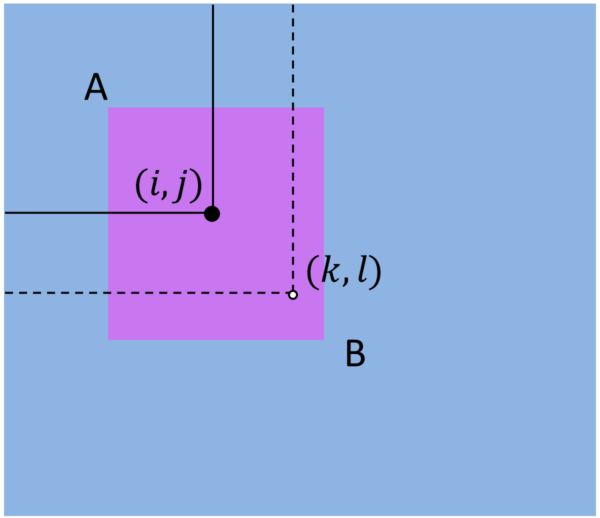


Linear Spatial Filter

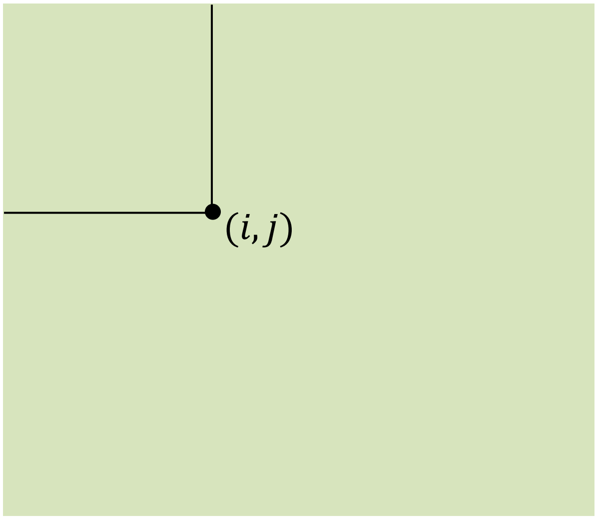
$$I'(u,v) \leftarrow \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j)$$

- $H(i,j)$

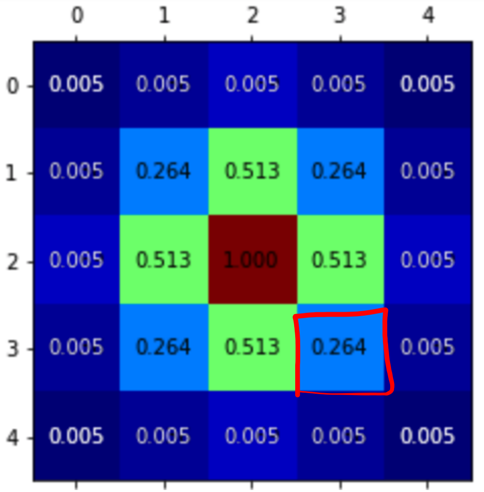
f



g

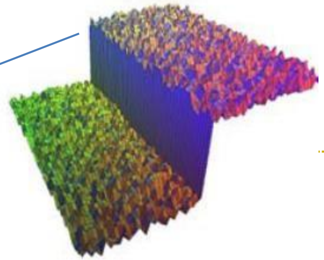


$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right)$$



$$g(i,j) = \frac{\sum_{k,l} f(k,l)\, d(i,j,k,l)}{\sum_{k,l} d(i,j,k,l)}$$

Bilateral Filter

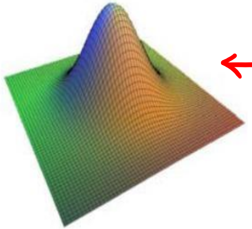


$f(x, y)$

$$g(i,j) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}. \tag{3.34}$$

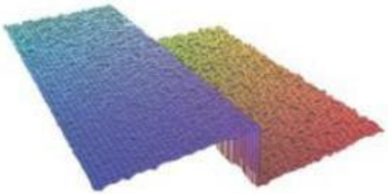
The weighting coefficient $w(i,j,k,l)$ depends on the product of a *domain kernel* (Figure 3.19c),

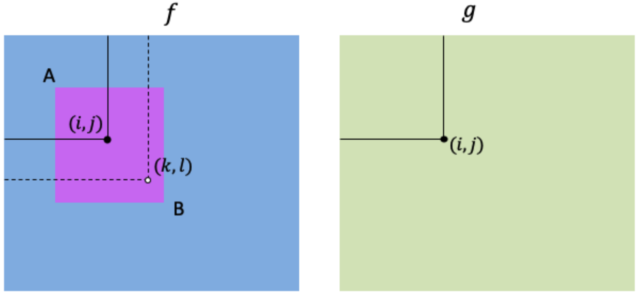
$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right),$$

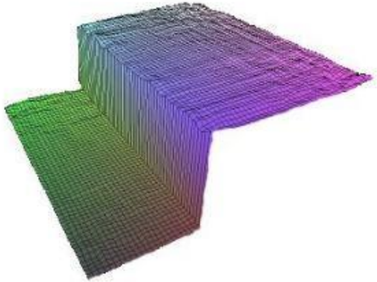


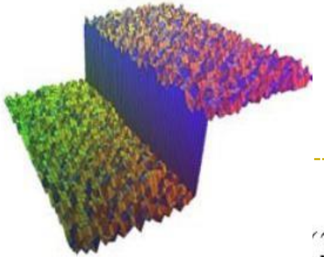
and a data-dependent *range kernel* (Figure 3.19d),

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right).$$

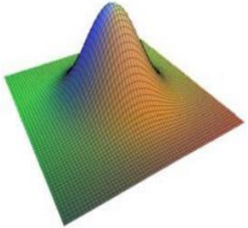






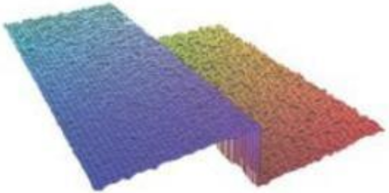


$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}. \tag{3.34}$$



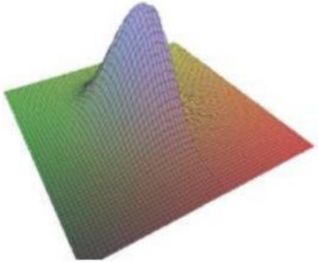
The weighting coefficient $w(i,j,k,l)$ depends on the product of a *domain kernel* (Figure 3.19c),

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right), \tag{3.35}$$



and a data-dependent *range kernel* (Figure 3.19d),

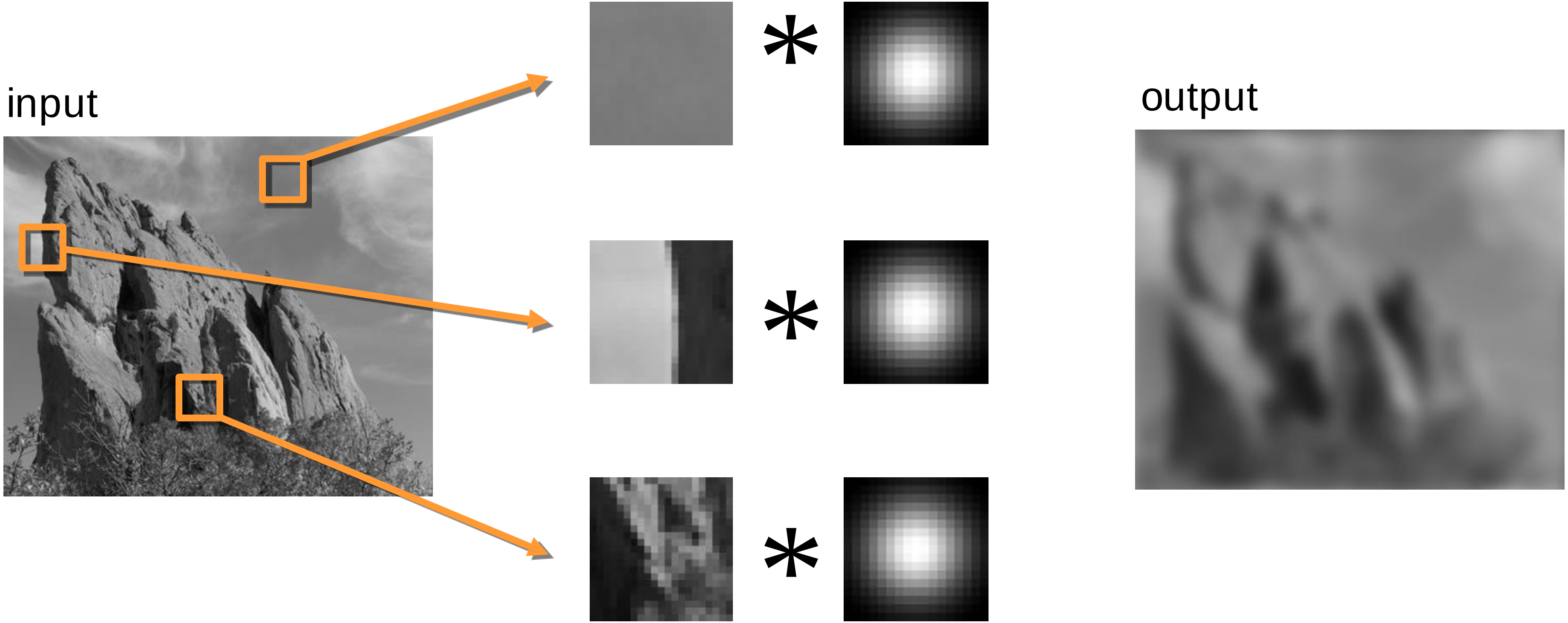
$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right).$$



When multiplied together, these yield the data-dependent *bilateral weight function*

$$\boxed{w(i,j,k,l)} = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right). \tag{3.37}$$

Usual Gaussian Filtering

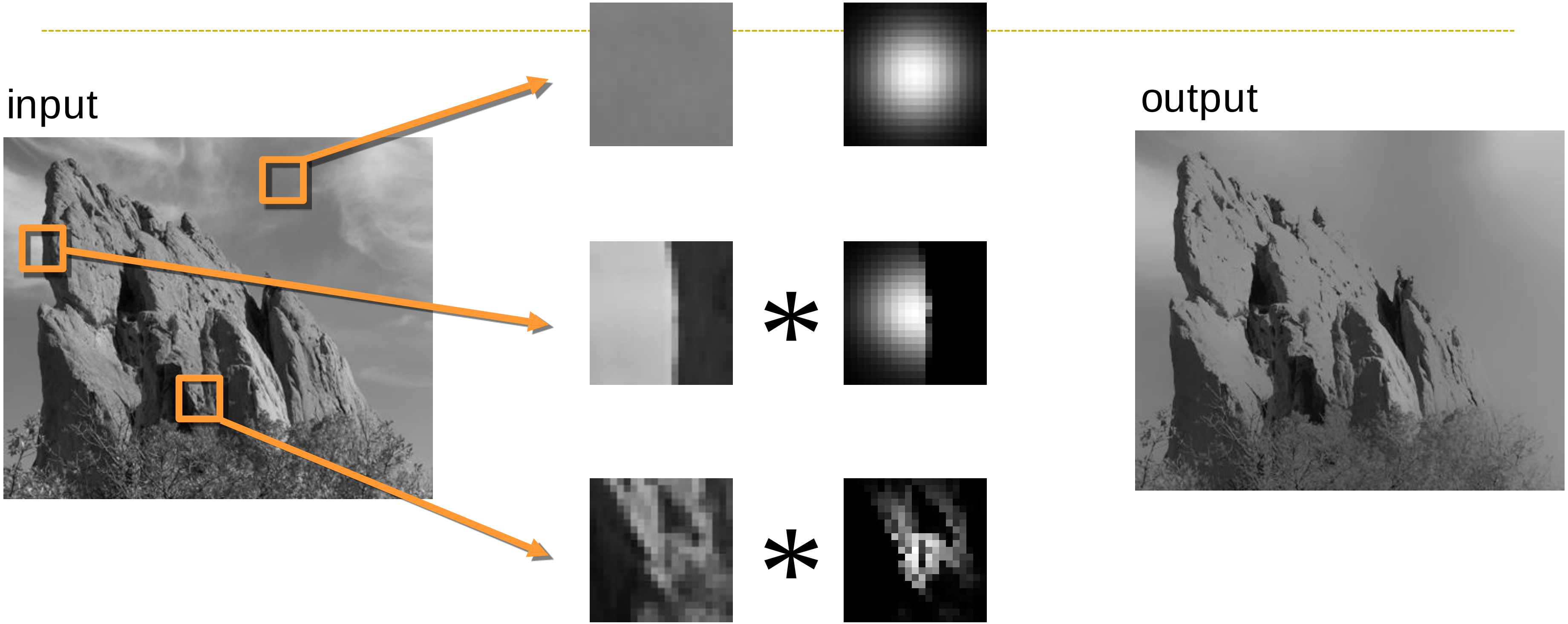


Same Gaussian kernel everywhere.

Source: Sylvain Paris

Bilateral Filtering

Bilateral Filter



The kernel shape depends on the image content.

Source: Sylvain Paris

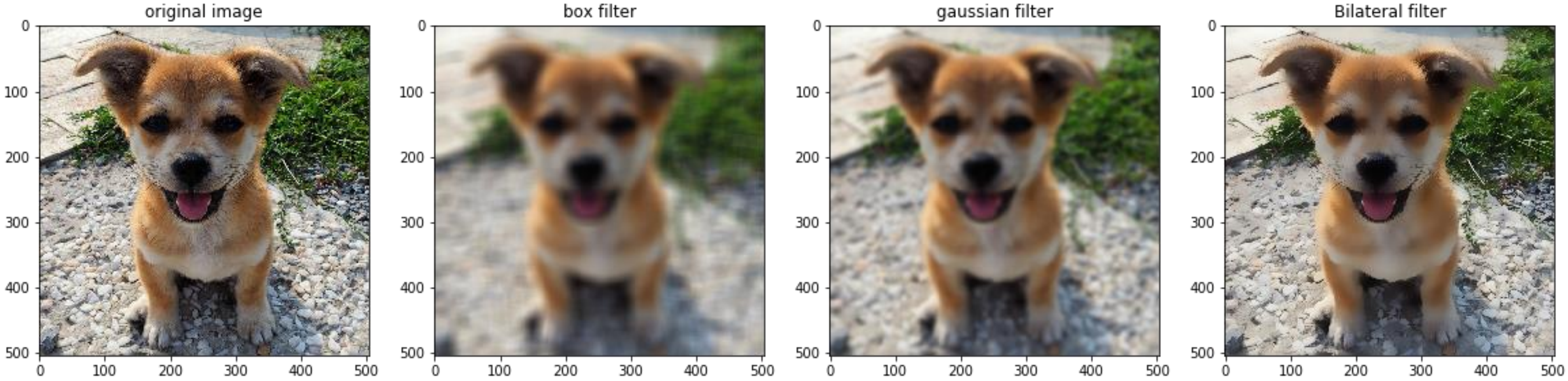
Fig. 2.4 Iterations: the bilateral filter can be applied iteratively, and the result progressively approximates a piecewise constant signal. This effect can help achieve a limited-palette, cartoon-like rendition of images [72]. Here, $\sigma_s = 8$ and $\sigma_r = 0.1$.

$\sigma_c \downarrow$ $\rightarrow \sigma_s$

$I \uparrow$ spatial $\uparrow$

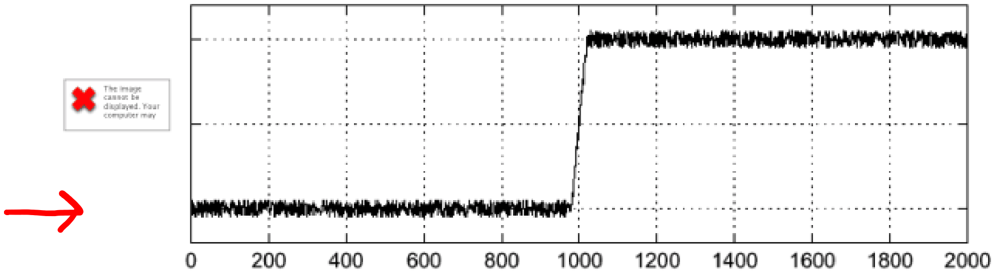
$\underbrace{\sigma_s, \sigma_c}$

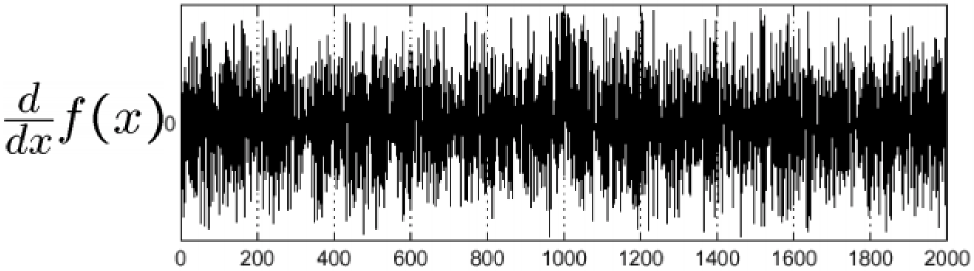
mean



Effects of noise

Consider a single row or column of the image

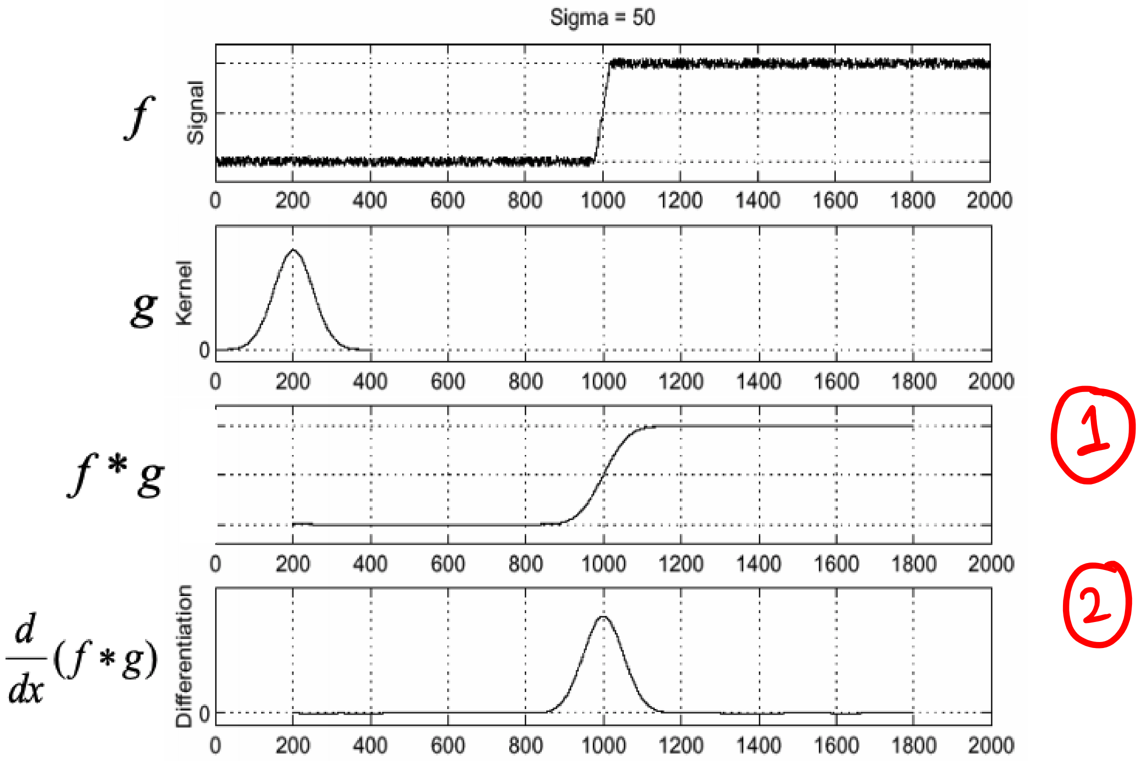




Where is the edge?

Source: S. Seitz

Solution: smooth first



$g(x)$

$g'(x)$

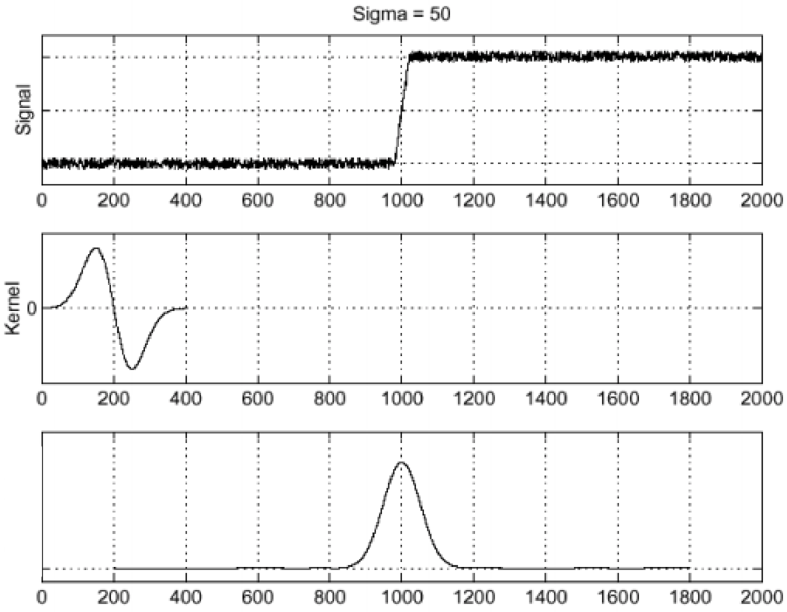
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

$\frac{d}{dx} g$

$f * \frac{d}{dx} g$



1

Source: S. Seitz

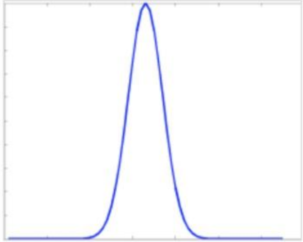
Other Important Filters

- Laplacian of Gaussian
  - Noise Suppression

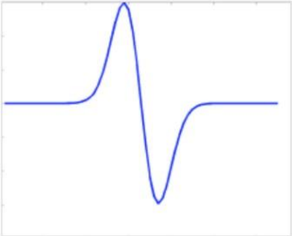
Robert Collins
CSE486

1D Gaussian and Derivatives

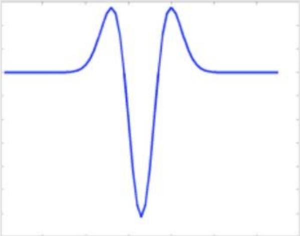
$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

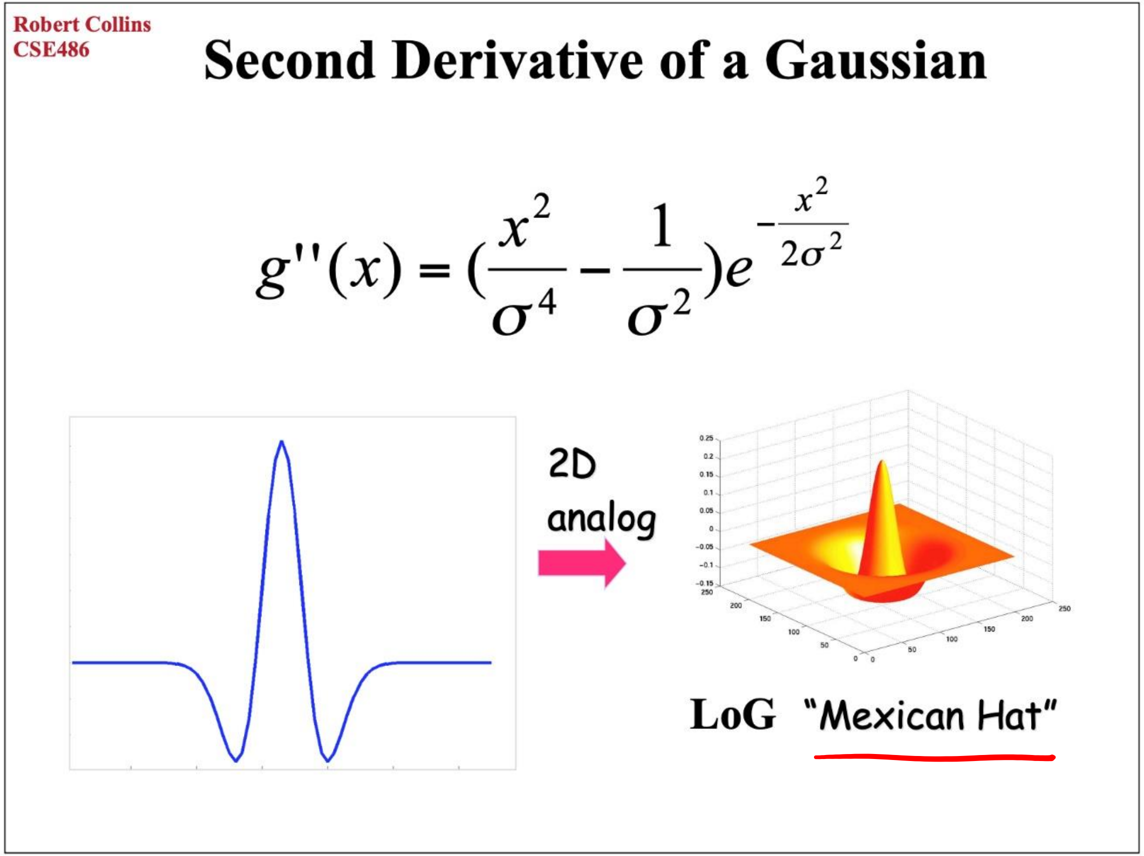


$$g''(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}}$$



Other Important Filters

- Laplacian of Gaussian
  - Noise Suppression



Other Important Filters

- Laplacian of Gaussian
  - Noise Suppression
- Difference of Gaussian
  - Band-pass

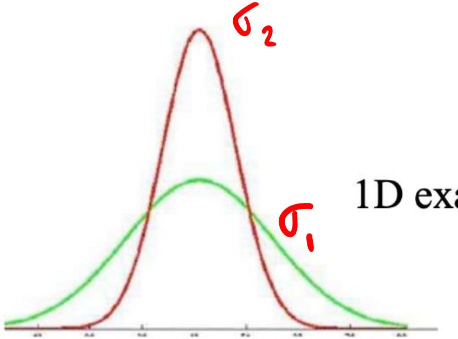
Robert Collins
CSE486

Efficient Implementation
Approximating LoG with DoG

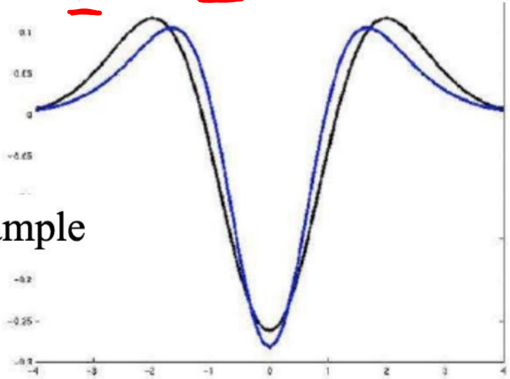
LoG can be approximate by a Difference of two Gaussians (DoG) at different scales

$$\nabla^2 G_\sigma \approx \boxed{G_{\sigma_1} - G_{\sigma_2}}$$

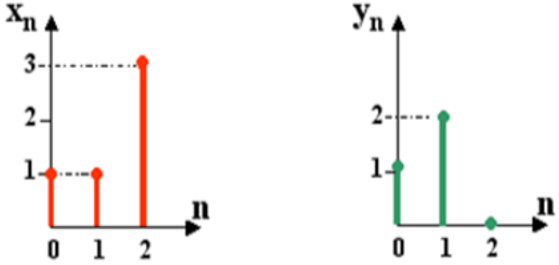
Best approximation when:
$\sigma_1 = \frac{\sigma}{\sqrt{2}}$, $\sigma_2 = \sqrt{2}\sigma$

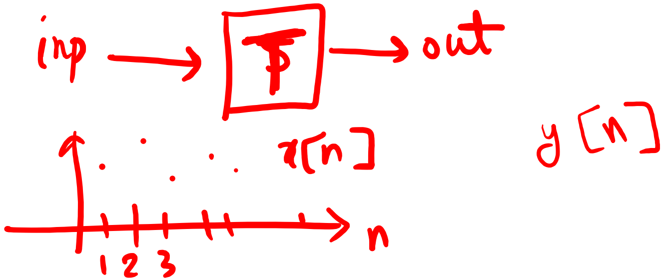


1D example



Linear System





A system T is **linear** if it satisfies the following two properties:

1) Scaling

$$x[n] \longrightarrow \boxed{T} \longrightarrow y[n] \quad \Rightarrow \quad \alpha\, x[n] \longrightarrow \boxed{T} \longrightarrow \alpha\, y[n]$$

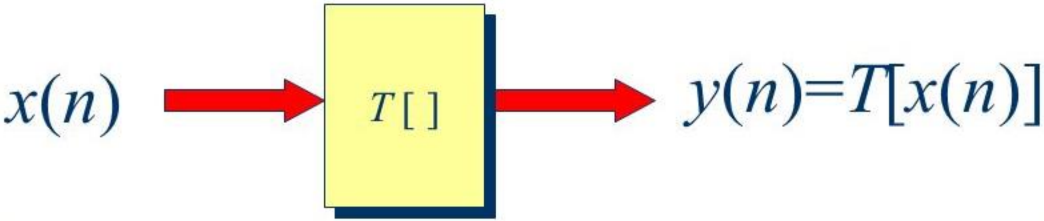
2) Additivity

$$x_1[n] \longrightarrow \boxed{T} \longrightarrow y_1[n] \qquad x_2[n] \longrightarrow \boxed{T} \longrightarrow y_2[n]$$

$$\Rightarrow \quad x_1[n] + x_2[n] \longrightarrow \boxed{T} \longrightarrow y_1[n] + y_2[n]$$

‘Linear’ Spatial Filtering

Linear System



$$T[ax_1(n)+bx_2(n)] = aT[x_1(n)]+bT[x_2(n)]$$

Convolution / Linear Filters

$I * f = I'$

correlation

- Smoothing (Average, Gaussian)
- Edge Filters (Prewitt, Sobel, Laplacian)

i \ j	-1	0	1
-1	a	b	c
0	d	e	f
1	g	h	i

$$I'(u,v) \leftarrow \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j) \boxed{\cdot\, H(i,j)}$$

averaging

$T[x[n]] = y[n]$

120	190	140	150	200
17	21	30	8	27
89	123	150	73	56
10	178	140	150	18
190	14	76	69	87

x

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

	98			

$x[n]$

$\boxed{\text{convolution}}$

References

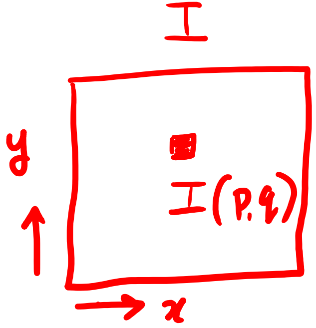
- GW Chapter – 3.4
- Convolution:

\- http://www.songho.ca/dsp/convolution/convolution.html

\- http://www.ceri.memphis.edu/people/smalley/ESCI7355/Ch6_Linear_Systems_Conv.pdf

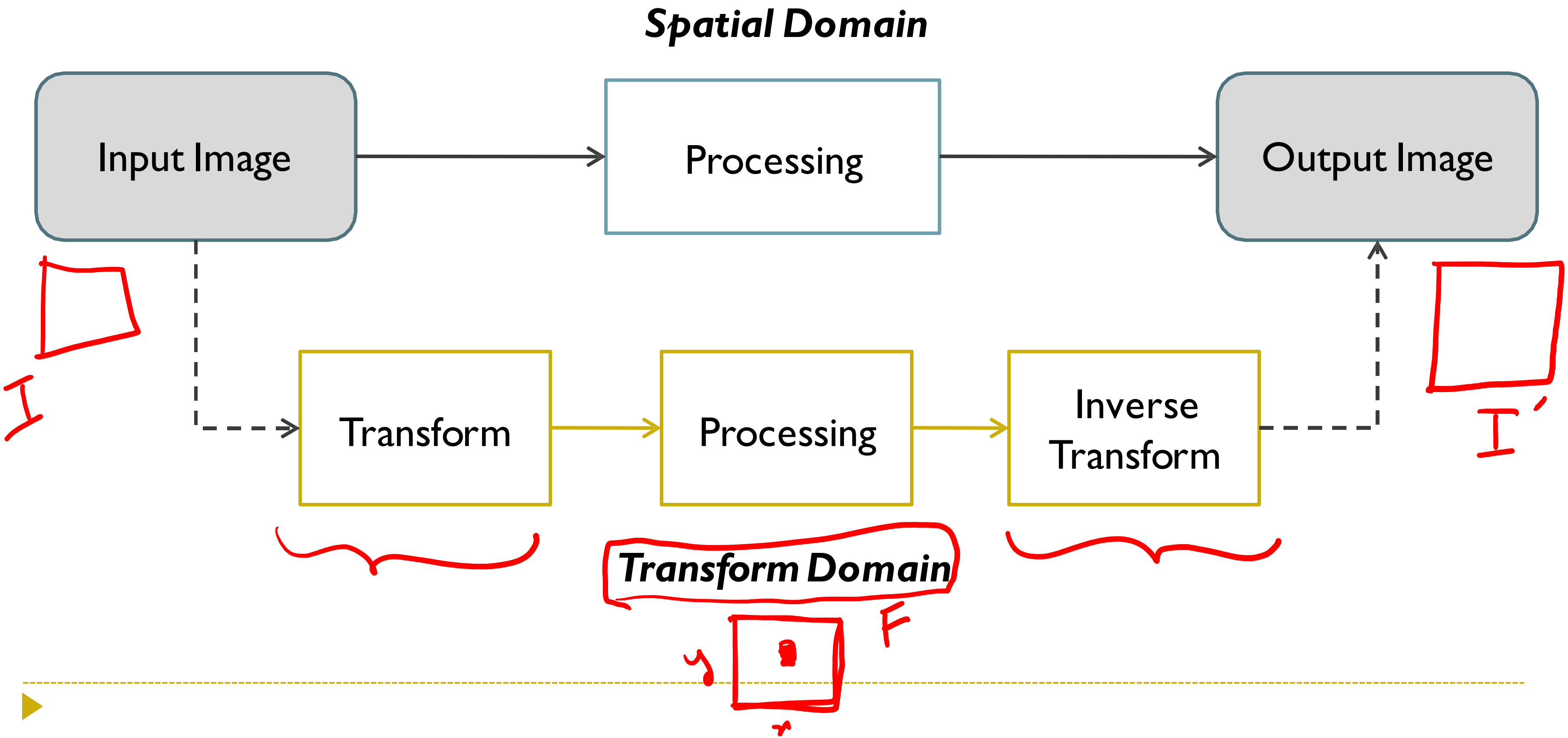
Image Processing – Two Paradigms

- Directly manipulating pixels in spatial domain



- Manipulating in 'transform domain'

Spatial vs. Transform Domain Processing



Spatial vs. Transform Domain Processing

Bandhani / Bandhej

Tie Dye

Spatial vs. Transform Domain Processing

Transform (Tie)

Process (Dye)

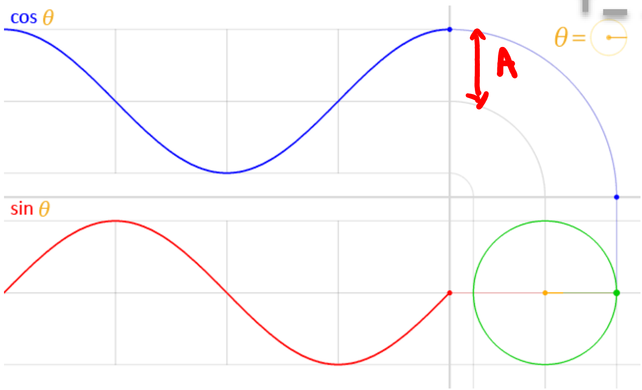
Inverse Transform (Untie)

Image Enhancement in Frequency Domain – Preliminary Concepts

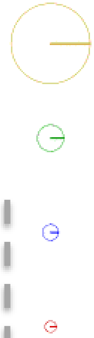
Periodic Signals

- Periodic → **Frequency** of occurrence
  - Repetitions/<Unit> ($\boxed{\text{cycles}}$/sec = Hz)

x(t) ↑ 0



A



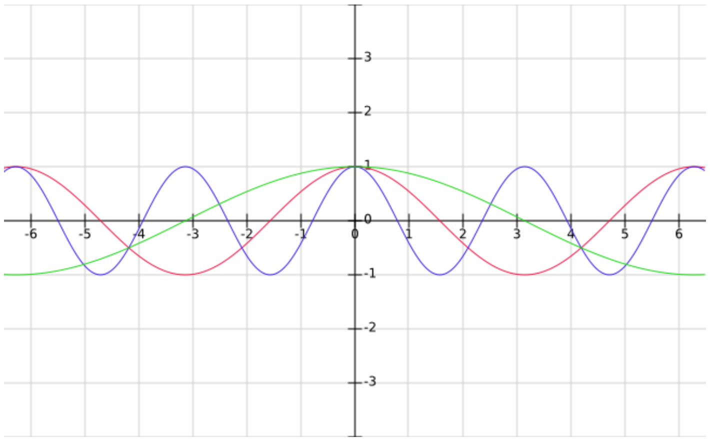
2π

$$x(t) = A\cos(\omega t) = A\cos(\boxed{2\pi} f t) = A\cos(\frac{2\pi}{T} t)$$

Angular frequency $\omega = 2\pi f$ Fundamental Period $f = \frac{1}{T}$

Simple periodic signals

- $x(t) = A\cos(t)$
- $x(t) = A\cos(2t)$
- $x(t) = A\cos(t/2)$

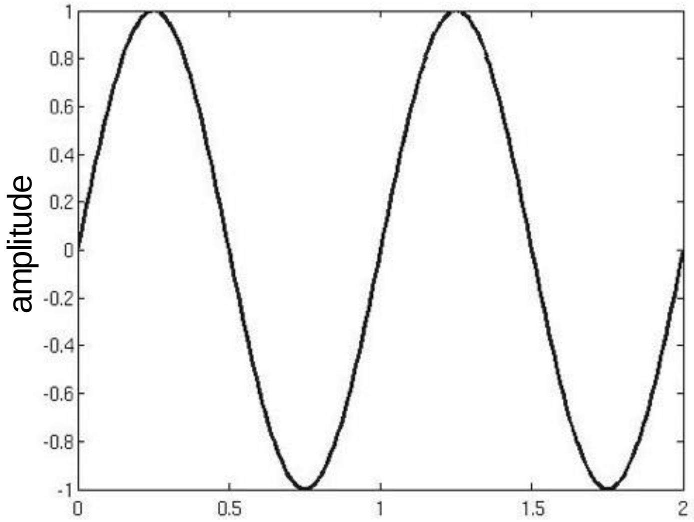


- $x(t) = A\cos(\omega t) = A\,cos(2\pi f t) = A\,cos(\frac{2\pi}{T}t)$

Angular frequency

Signal and Frequency Domains

$$\boxed{y = \sin(2\pi t)}$$

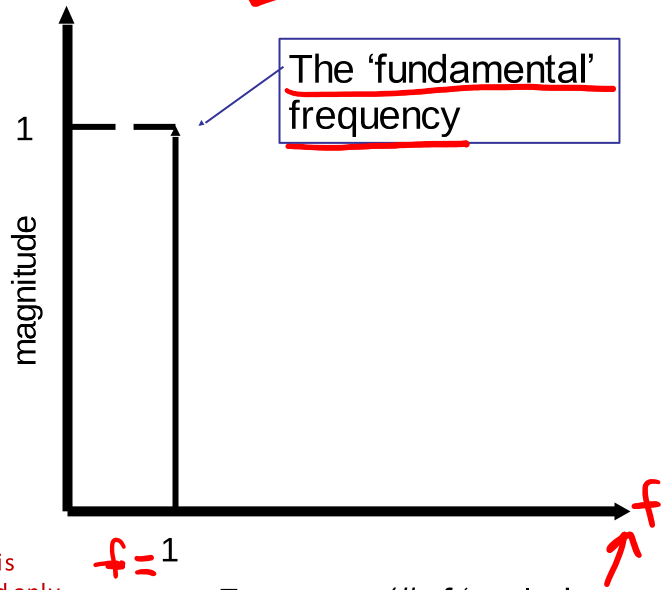


NOTE: t=0 is time from where signal is considered

time/length

Signal domain

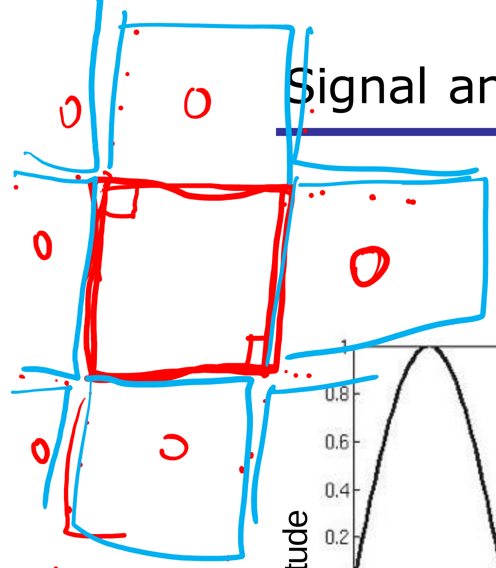
NOTE: t is bounded only for physical signals



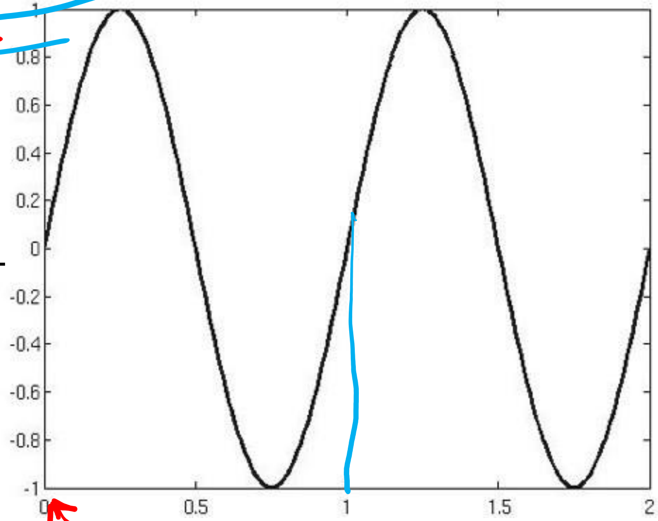
Frequency (# of 'cycles' per unit time)

Frequency domain

Signal and Frequency Domains



$$y = \sin(2\pi s)$$

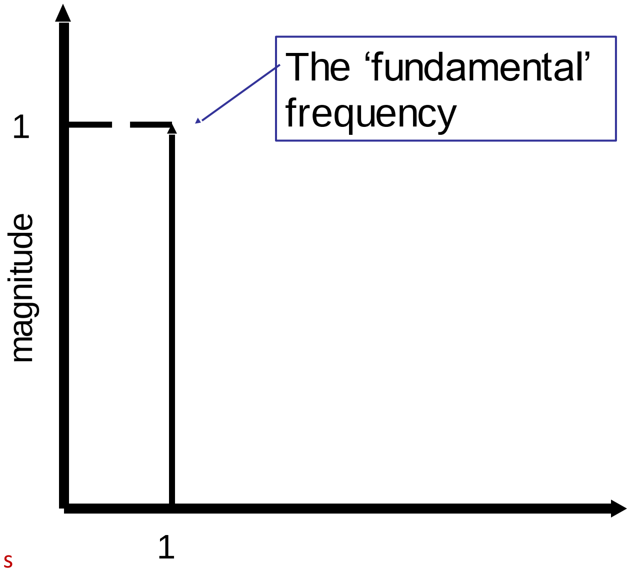


NOTE: s=0 is location from where signal is considered

spatial distance (s)

Signal domain

NOTE: s is bounded only for physical signals



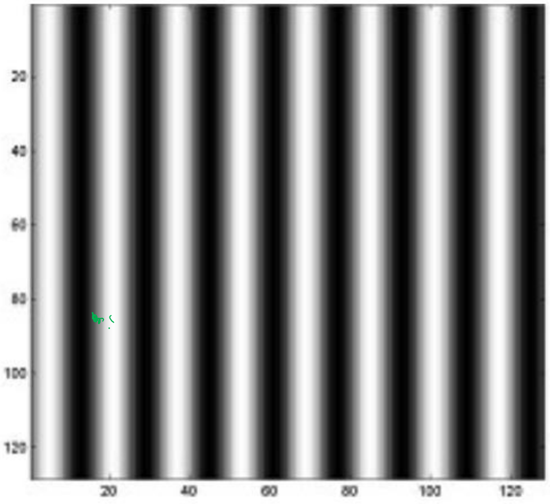
Frequency (# of cycles per unit distance)

Frequency domain

Periodic Images

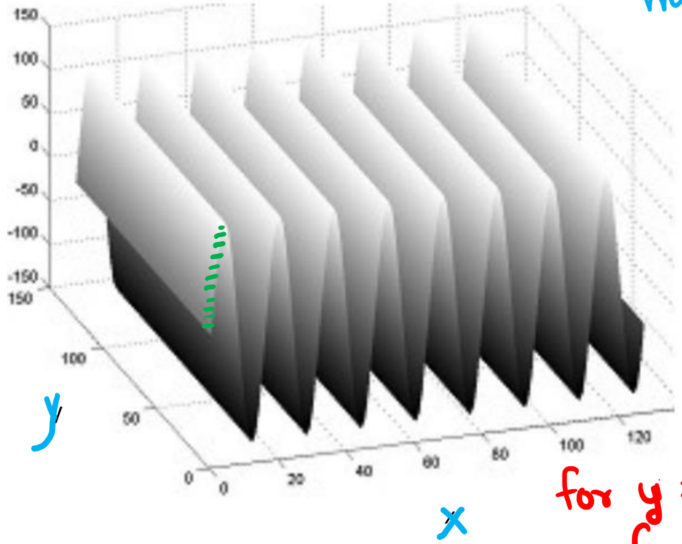
spatial signal

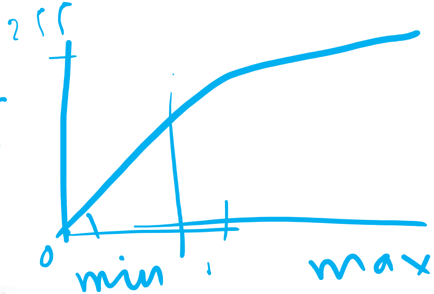
128 x 128 ~~grayscale image~~



Sinusoid pattern repeats every 16 pixels
f = 1/16 cycles/pixel

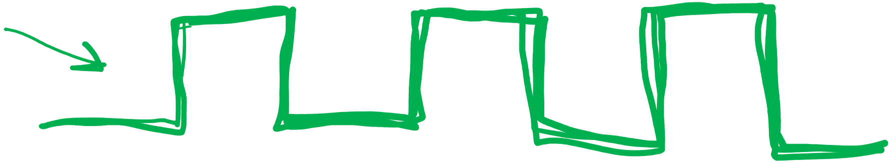
$$I(x, y) = 128\, sin(2\pi x/16)$$





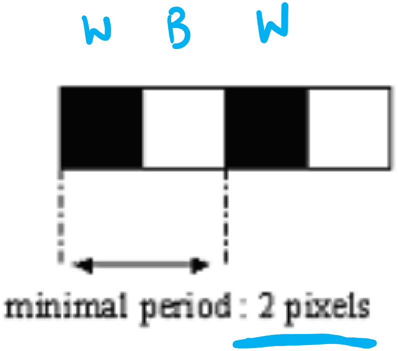
128 $\frac{1}{16}$

for y = 1:128
for x=1:128
128 sin(2π x/16)

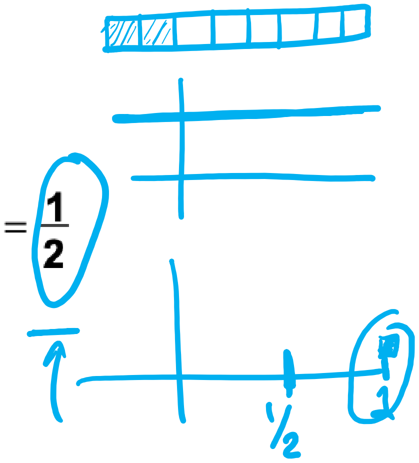


Periodic Images

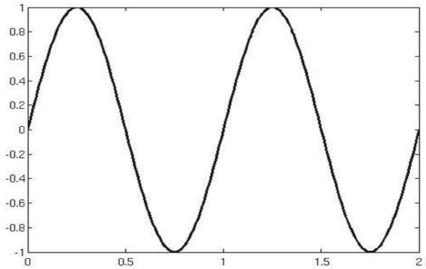
Spatial period = Minimal # of pixels between two identical patterns in a “periodic” image



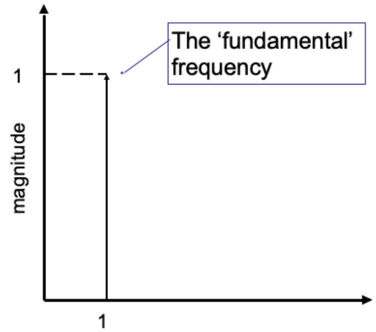
$$\Rightarrow \quad \nu_{max} = \frac{1}{\text{minimal period}} = \frac{1}{2}$$



$y = \sin(2\pi s)$

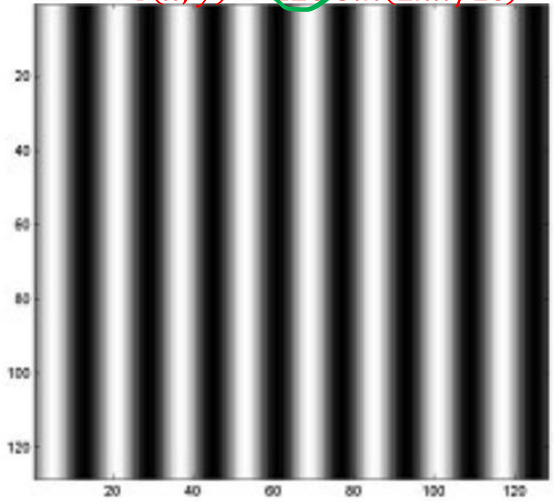


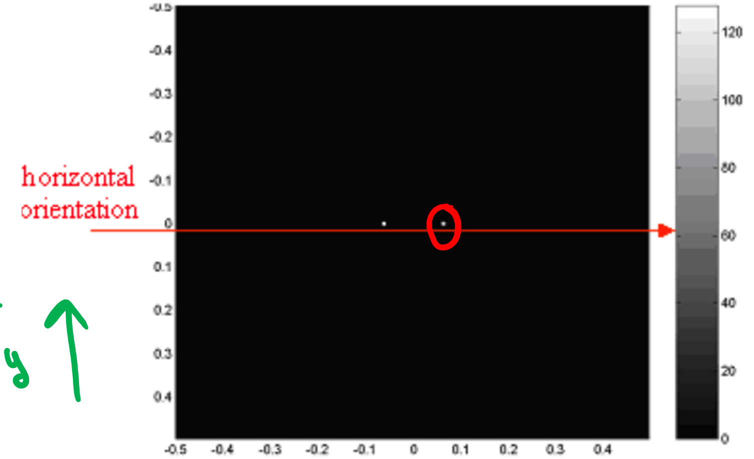
spatial distance (s)

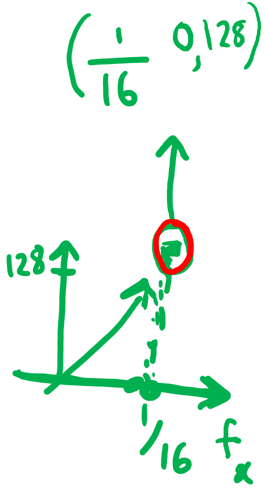


Frequency (# of cycles per unit distance)

$I(x, y) = 128 \sin(2\pi x / 16)$







f_y

f_x

Sinusoid pattern repeats every 16 pixels

f = 1/16 cycles/pixel

Spatial domain

Frequency domain

Scribe List

2018101055
2018101058
2018101059
2018101066
2018101069
2018101070