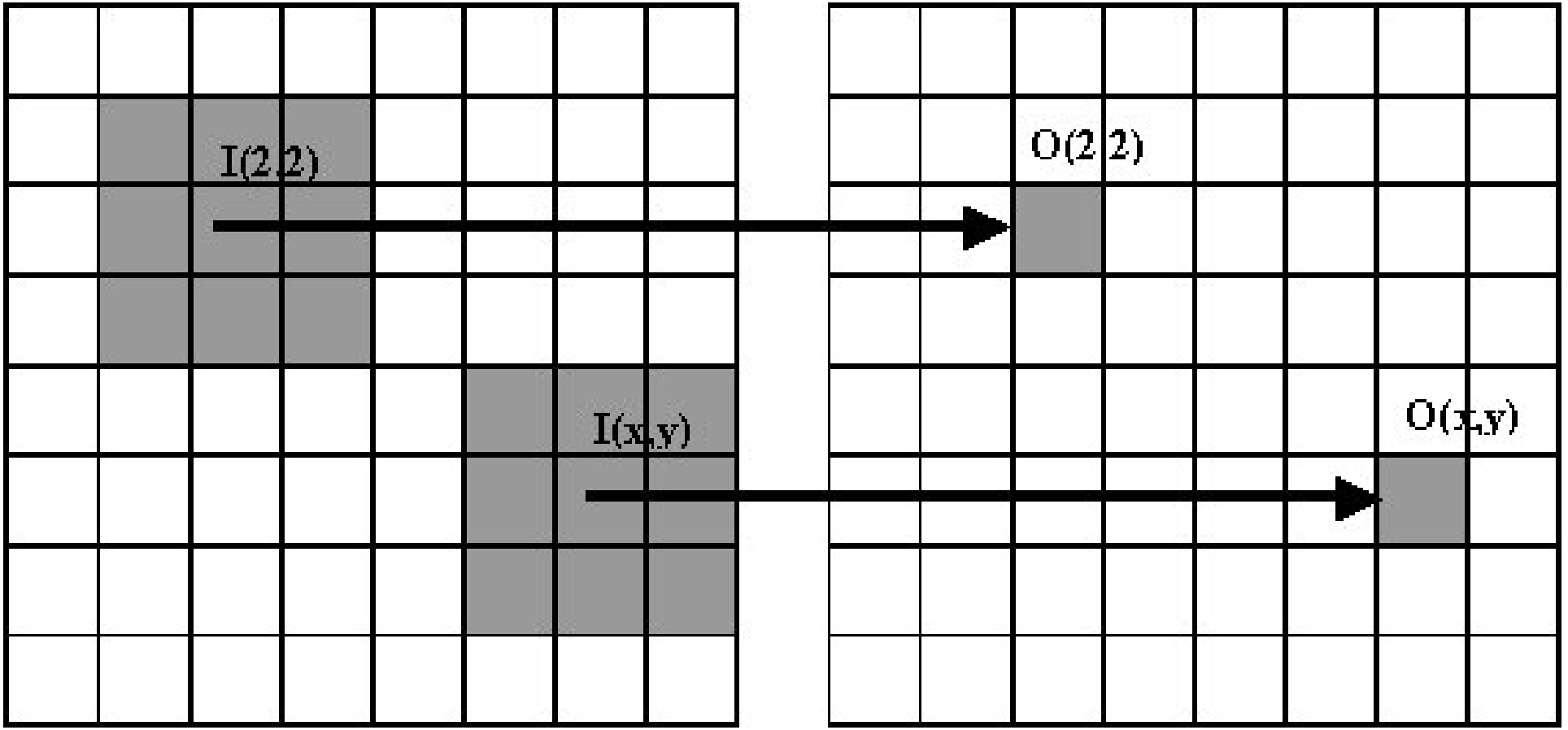
Digital Image Processing

Image Enhancement : Spatial Filtering

Suyash P. Awate

Spatial Filtering

- Key idea
  - Intensity at pixel “p” in output image is a function of the pixel intensities in the neighborhood of “p” in input image



Spatial Filtering

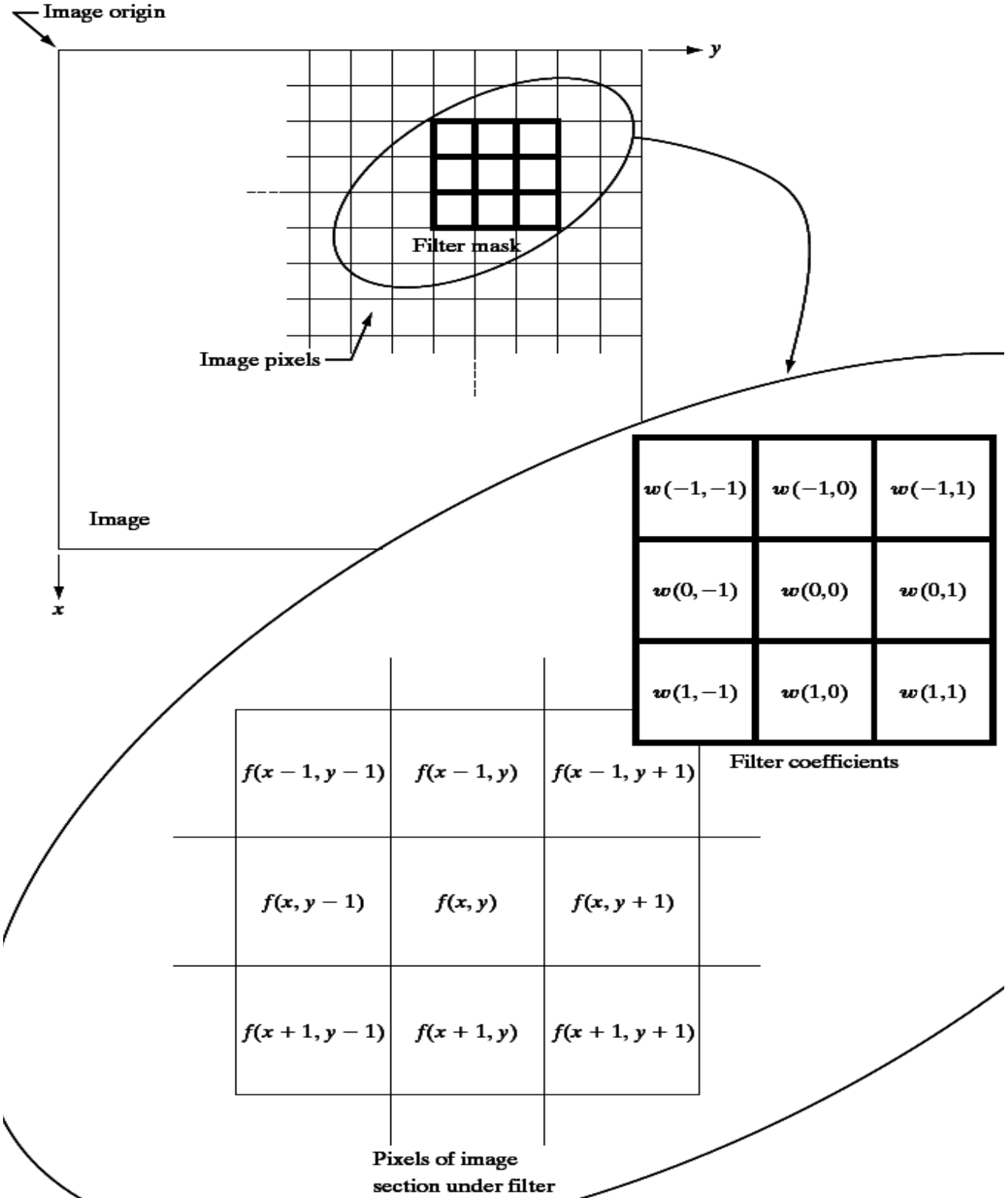
- **Linear** spatial filtering
  - Spatial filtering where the **output** pixel intensity = **linear function** of the **input** pixel intensities

Spatial Filter

- Linear spatial filter
  - g (x,y) typically designed as =

    $\sum_{i=-1,0,1} \sum_{j=-1,0,1}$

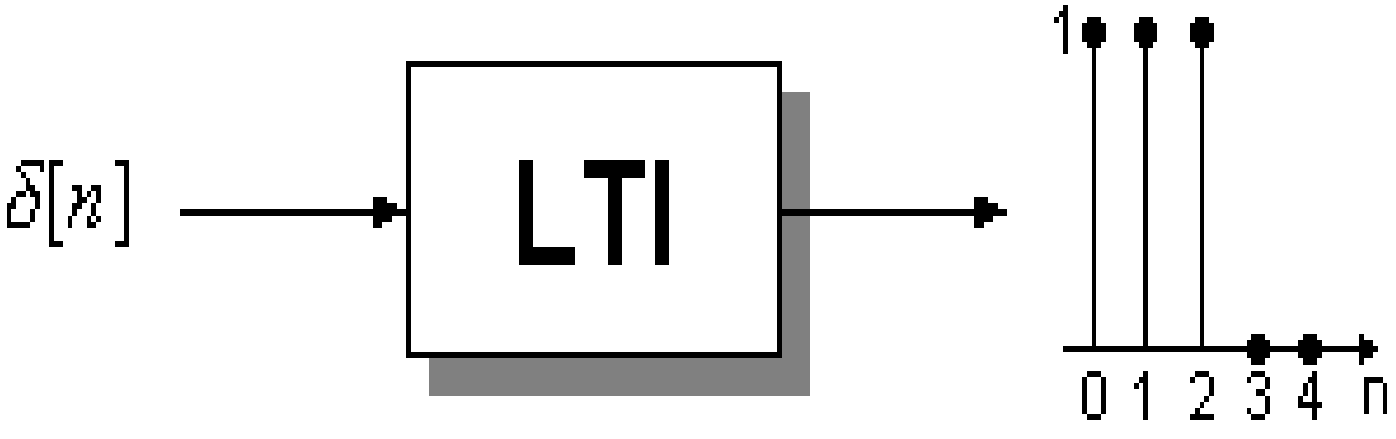
    w (i,j) * f (x+i,y+j)
- w = **filter mask**
- w(i,j) = filter **coefficients**
- Linear spatial filter performs "**convolution**"



Spatial Filtering

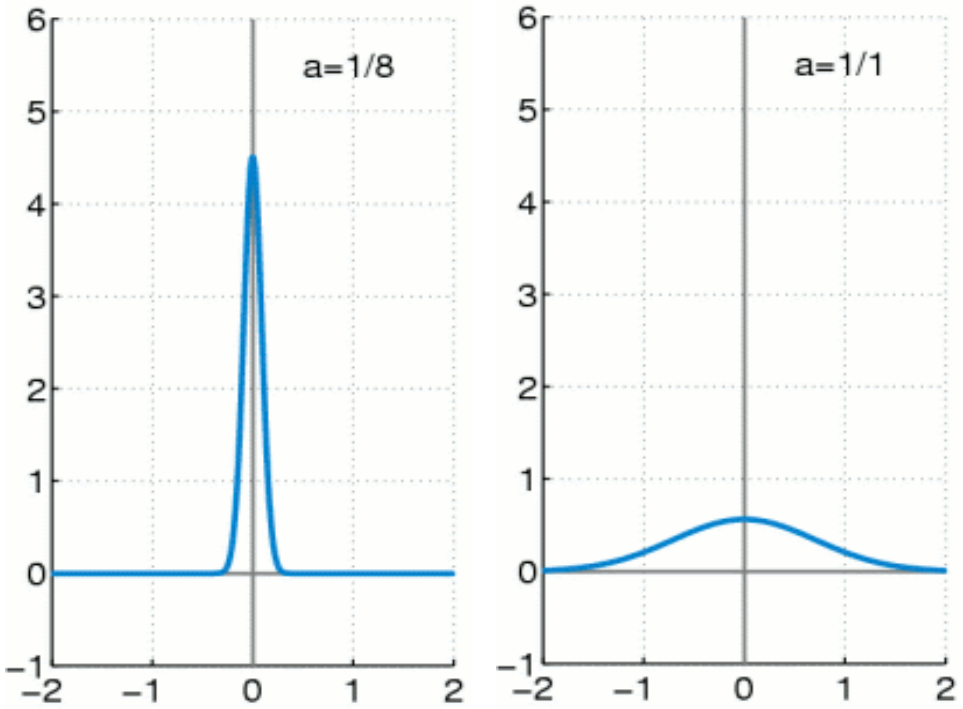
- Consider a black-box system
  - Takes input x(t) → Produces output y(t)

  - Let input = **(discrete) unit-impulse function** at time t=0
    - **Kronecker delta**: δ(t) = 1 if t =0; Otherwise δ(t) = 0
    - **Continuous** impulse function = **Dirac delta**
      - Derivative of the unit-step function
      - Rectangular pulse with width → 0, height → ∞, area = 1
      - Limit of a Gaussian PDF as variance → 0
  - For input = δ(t), output = h(t) = **impulse response**



Spatial Filtering

- Continuous impulse function (Dirac delta)
  - Definition : $\delta_a(x) = \frac{1}{a\sqrt{\pi}} e^{-x^2/a^2}$ as $a \to 0$.

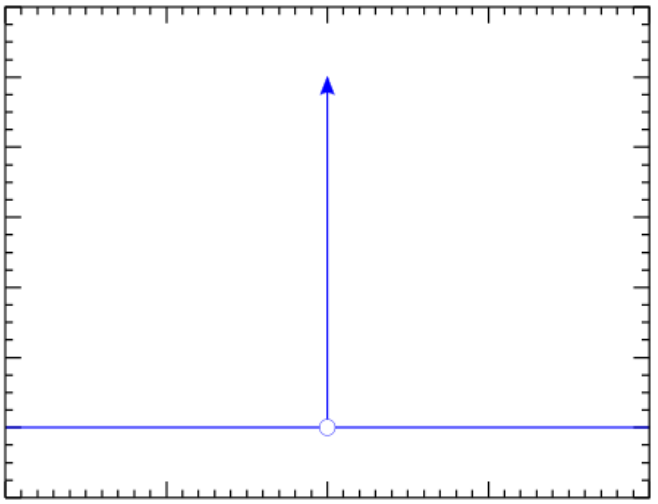


    - δ(x) = 0 for all |x| > 0
    - δ(x) integrates to 1

$$\int_{-\infty}^{\infty} \delta(x)\, dx = 1$$

  - Sifting / sampling property

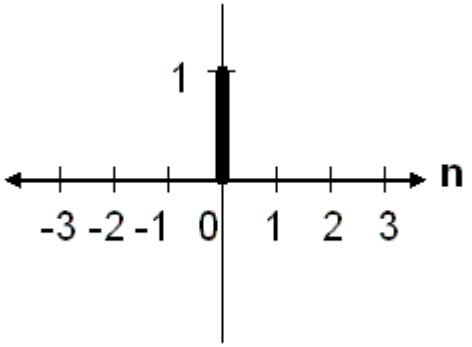
$$\int_{-\infty}^{\infty} f(t)\delta(t-T)\, dt = f(T)$$



Spatial Filtering

- Discrete impulse function (Kronecker delta)
  - Definition :

$$\delta[n] = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases}$$

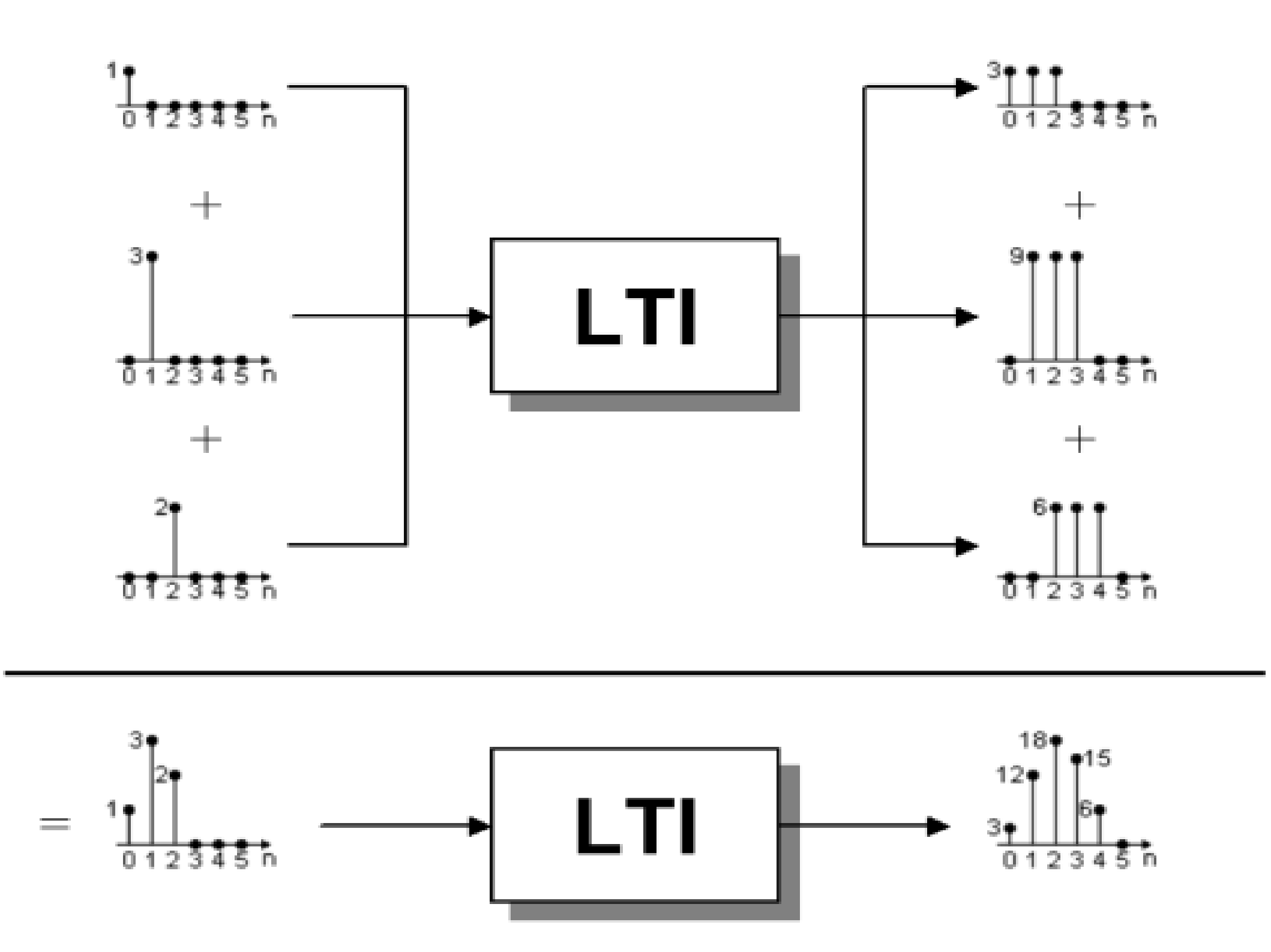


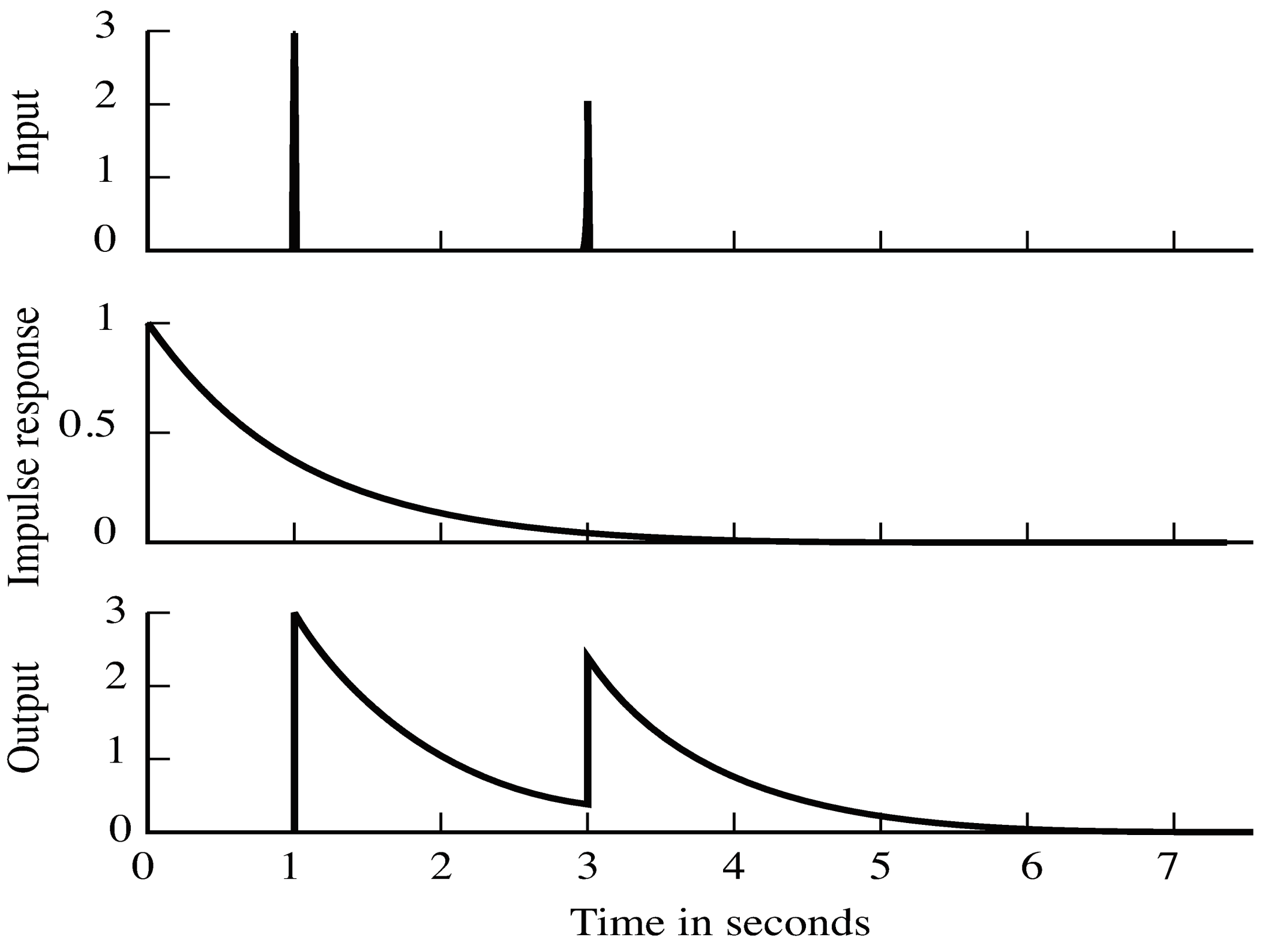
  - Sifting property

$$\sum_{i=-\infty}^{\infty} a_i \delta_{ij} = a_j$$

Spatial Filtering

- Input-output system
  - For a time-invariant system:
    - When input gets **delayed** / **shifted** by some time interval, output gets shifted by the same time interval
      - $\delta(t-t') \rightarrow h(t-t')$
  - For a linear system:
    - When input gets **scaled** by a factor, output gets scaled by same factor
      - $a\,\delta(t) \rightarrow a\,h(t)$
    - When the input is a **sum** of 2 (shifted) impulse functions, the output is the sum of 2 (shifted) impulse responses
      - $\delta(t) + \delta(t-t') \rightarrow h(t) + h(t-t')$
    - See picture on next slide ...


1
0 1 2 3 4 5 n
3
0 1 2 3 4 5 n
2
0 1 2 3 4 5 n
LTI
3
0 1 2 3 4 5 n
9
0 1 2 3 4 5 n
6
0 1 2 3 4 5 n
=
3
2
1
0 1 2 3 4 5 n
LTI
18
12
15
6
3
0 1 2 3 4 5 n


Input
3
2
1
0
Impulse response
1
0.5
0
Output
3
2
1
0
0
1
2
3
4
5
6
7
Time in seconds

Spatial Filtering

- Linear time-invariant system
  - f (shifted candle)
    = shifted (f (candle))
  - f (candle1 + candle2)
    = f (candle1) + f (candle2)

Spatial Filtering

- **Linear time-invariant system**
  - In equations:
    - Impulse response = H(t)
    - Input = F(t)
    - Output = G(t)
  - We can **rewrite any input F(t)** as

$$\int_{-\infty}^{\infty} f(t)\delta(t-T)\,dt = f(T)$$

    - F(t) = F(0) δ(t) + F(1) δ(t-1) + F(2) δ(t-2) + …
  - By **definition of impulse response**,
    if input = δ(t),
    then output = H(t)
  - By **definition of LTI system**,
    if input is F(t) = sum of scaled shifted δ(t) functions,
    then output G(t) = sum of scaled shifted H(t) responses

Spatial Filtering

- **Linear time-invariant system**
  - By the definition of the LTI system,
    output G(t) = sum of scaled shifted impulse responses
    - $G(t) = F(0)\,H(t) + F(1)\,H(t-1) + F(2)\,H(t-2) + \ldots$
    - $G(t) = \sum_{k=0,1,2} F(k)\,H(t-k)$
  - Output G(t)
    = **convolution** of input F(t) with impulse response H(t)
  - In general
    - Input function can be arbitrarily long
    - Impulse response function can be arbitrarily long
    - **$G(t) = \sum_{k=-\infty,\ldots,\infty} F(k)\,H(t-k)$**
  - **$G = F * H$**
  - **$G(t) = (F * H)(t)$**

Spatial Filtering

- Convolution
  - What if impulse response H(t) = δ(t) ?
    - If, impulse response = impulse function, then F * H = ?

Spatial Filtering

- Convolution
  - Convolution is a **symmetric / commutative** operation
  - F * H = H * F
  - Observe

    $G(t) = (F * H)(t) = \sum_{k=-\infty,\ldots,\infty} F(k)\, H(t-k)$

    $= \sum_{m=-\infty,\ldots,\infty} F(t-m)\, H(m) = (H * F)(t)$

    - Just a change of variables:

      Define $m = t - k$

      Then,

      $k = t - m$

      $k = \infty \quad \rightarrow \quad m = (-\infty)$

      $k = (-\infty) \quad \rightarrow \quad m = \infty$

Spatial Filtering

- Convolution
  - Convolution is a **linear** operation on its arguments
  - What if H(t) = A(t) + B(t) ?
    - F * H = F * (A + B) = ?
  - What if H(t) = c A(t) ? where c = constant
    - F * H = F * (c A) = ?

Spatial Filtering

- Convolution
  - Convolution is **associative** (order doesn't matter)
    - (x * h1) * h2 = x * (h1 * h2)
  - Proof: Left hand side (t) = $\int_{-\infty}^{\infty} x(\tau)h_1(t-\tau)d\tau * h_2(t)$
  - (expanding) = $\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} x(\tau)h_1(\gamma-\tau)d\tau\right]h_2(t-\gamma)d\gamma$
  - (reverse order) = $\int_{-\infty}^{\infty} x(\tau)\left[\int_{-\infty}^{\infty} h_1(\gamma-\tau)h_2(t-\gamma)d\gamma\right]d\tau$
  - $\varphi = \gamma - \tau$
  - (substitution) = $\int_{-\infty}^{\infty} x(\tau)\left[\int_{-\infty}^{\infty} h_1(\varphi)h_2(t-\tau-\varphi)d\gamma\right]d\tau$
  - Inner integral = h1 * h2 evaluated at (t − τ)
  - Outer integral = x * (h1 * h2) evaluated at t

Spatial Filtering

- Convolution
  - How to **compute** convolution for discrete images ?
    - Consider time t = 0
      - We can redefine any arbitrary time-point as “0” by translating the coordinate frame for input F (and output G)
      - The origin of the time coordinate frame is NOT important in LTI systems
    - $G(t=0) = \sum_{k=-\infty,\ldots,\infty} F(k)\, H(0-k)$
    - What does this mean algorithmically ?
      - To compute G(0):

        (1) **Flip** the impulse response about the origin → H(-k)

        (2) Perform **pointwise multiplication** of F and flipped-H

        (3) **Sum** up
    - How to compute G(t) at some arbitrary timepoint “t” ?

Spatial Filtering

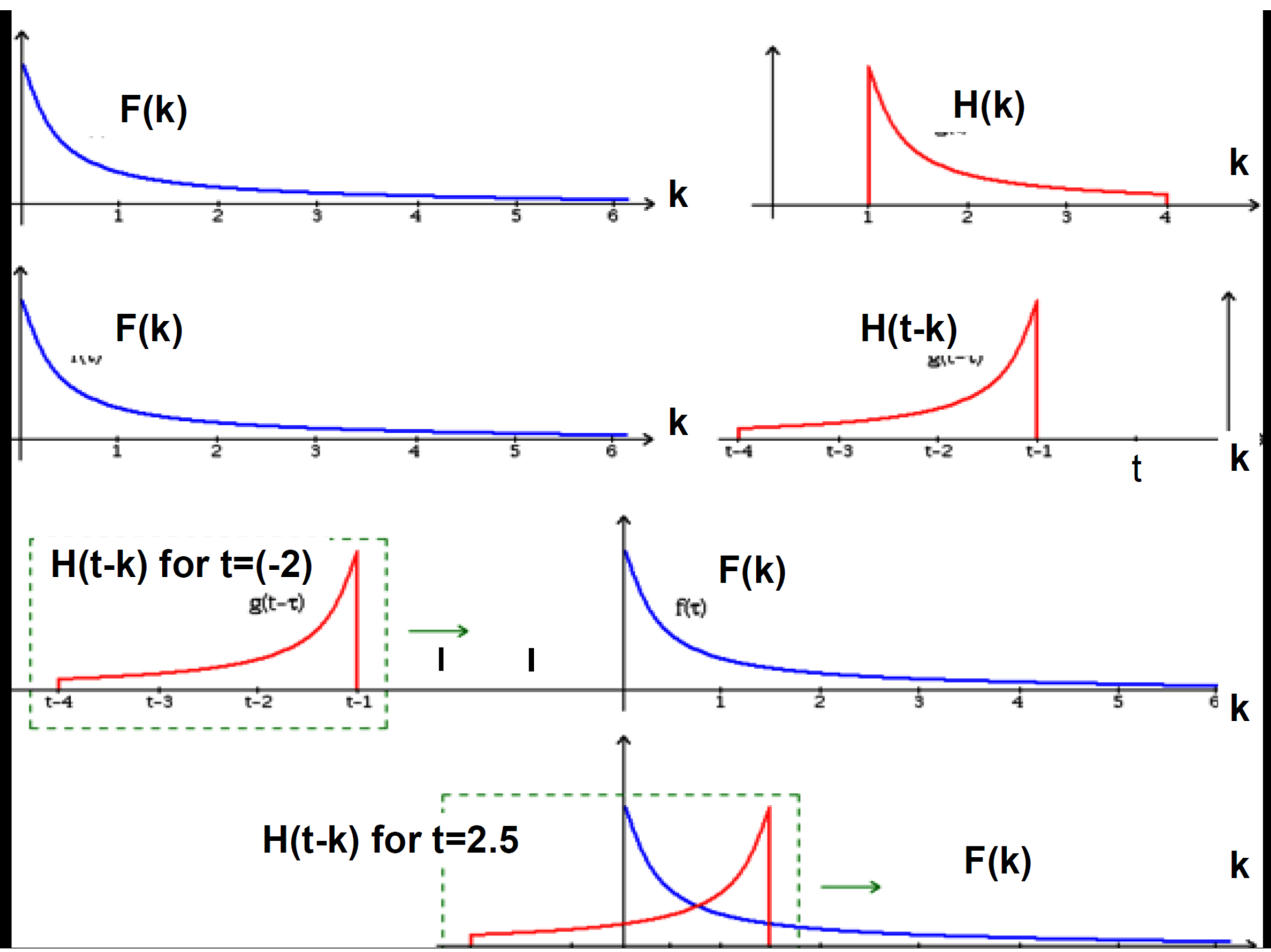
- Convolution
  - What does this mean algorithmically ?
    - We want to compute $G(t) = \sum_{k=-\infty,\ldots,\infty} F(k)\, H(t-k)$ for an arbitrary timepoint t

      (1) **Flip** the impulse response about the origin → H(-k)

      (2) **Shift** the flipped impulse response by t → H(t-k)

      (3) Perform **pointwise multiplication** of F and shifted-flipped-H

      (4) **Sum** up
  - See next slide for pictures


F(k)
k
1
2
3
4
5
6
H(k)
k
1
2
3
4
F(k)
k
1
2
3
4
5
6
H(t-k)
t-4
t-3
t-2
t-1
t
k
H(t-k) for t=(-2)
g(t-τ)
t-4
t-3
t-2
t-1
f(τ)
F(k)
1
2
3
4
5
6
k
H(t-k) for t=2.5
F(k)
k

Spatial Filteri

convolution

/ˌkɒnvəˈluːʃ(ə)n/

noun

noun: **convolution**; plural noun: **convolutions**; noun: **convolution integral**; plural noun: **convolution integrals**; plural noun: **convolutions integral**

1. a coil or twist.
   "crosses adorned with elaborate convolutions"
   synonyms: twist, turn, coil, spiral, twirl, curl, helix, whorl, loop, curlicue, kink, sinuosity; More
   - the state of being or process of becoming coiled or twisted.
     "the flexibility of the polymer chain allows extensive convolution"
2. a thing that is complex and difficult to follow.
   "the convolutions of farm policy"
   synonyms: complexity, intricacy, complication, twist, turn, entanglement, contortion; More
3. a sinuous fold in the surface of the brain.
4. MATHEMATICS
   a function derived from two given functions by integration which expresses how the shape of one is modified by the other.
   - a method of determination of the sum of two random variables by integration or summation.

Origin

mid 16th century: from medieval Latin *convolutio(n-*), from *convolvere* 'roll together' (see convolve).

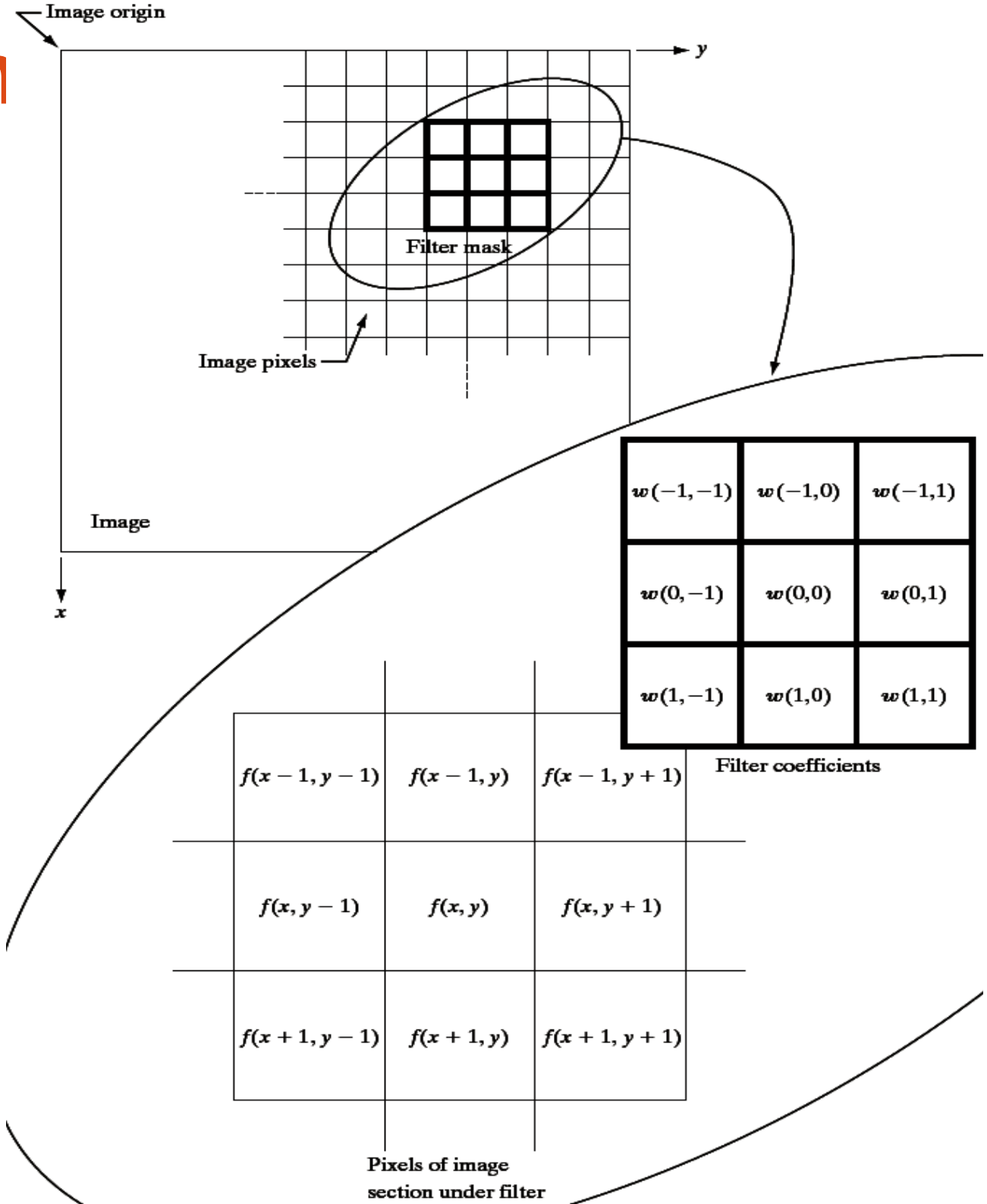
The sign was just as convoluted as the road.

Spatial Filterin

- Convolving images
  - Domain = spatial
  - System is **linear** **space**-invariant / **shift**-invariant
  - Impulse response = **point-spread function**
  - Typically, w(x,y) = w (-x,-y)
  - g (x,y) =

    $\sum_{i=-1,0,1} \sum_{j=-1,0,1}$

    w (i,j) * f (x+i,y+j)



Spatial Filtering

- **Cross-Correlation**
  - Signals: F(k) and H(k)
  - **Measure of similarity** of two signals / images
    - Cross-correlation of F and H is $\mathbf{G(0) = \sum_{k=-\infty,...,\infty} F(k)\, H(k)}$
  - Similarity between F and a shifted version of H
    - Cross-correlation of F and H is $\mathbf{G(t) = \sum_{k=-\infty,...,\infty} F(k)\, H(k+t)}$
  - What does this mean algorithmically ?

    (1) **Shift** one of the signals by t → H(k+t)

    (2) Perform **pointwise multiplication** of F and shifted-flipped-H

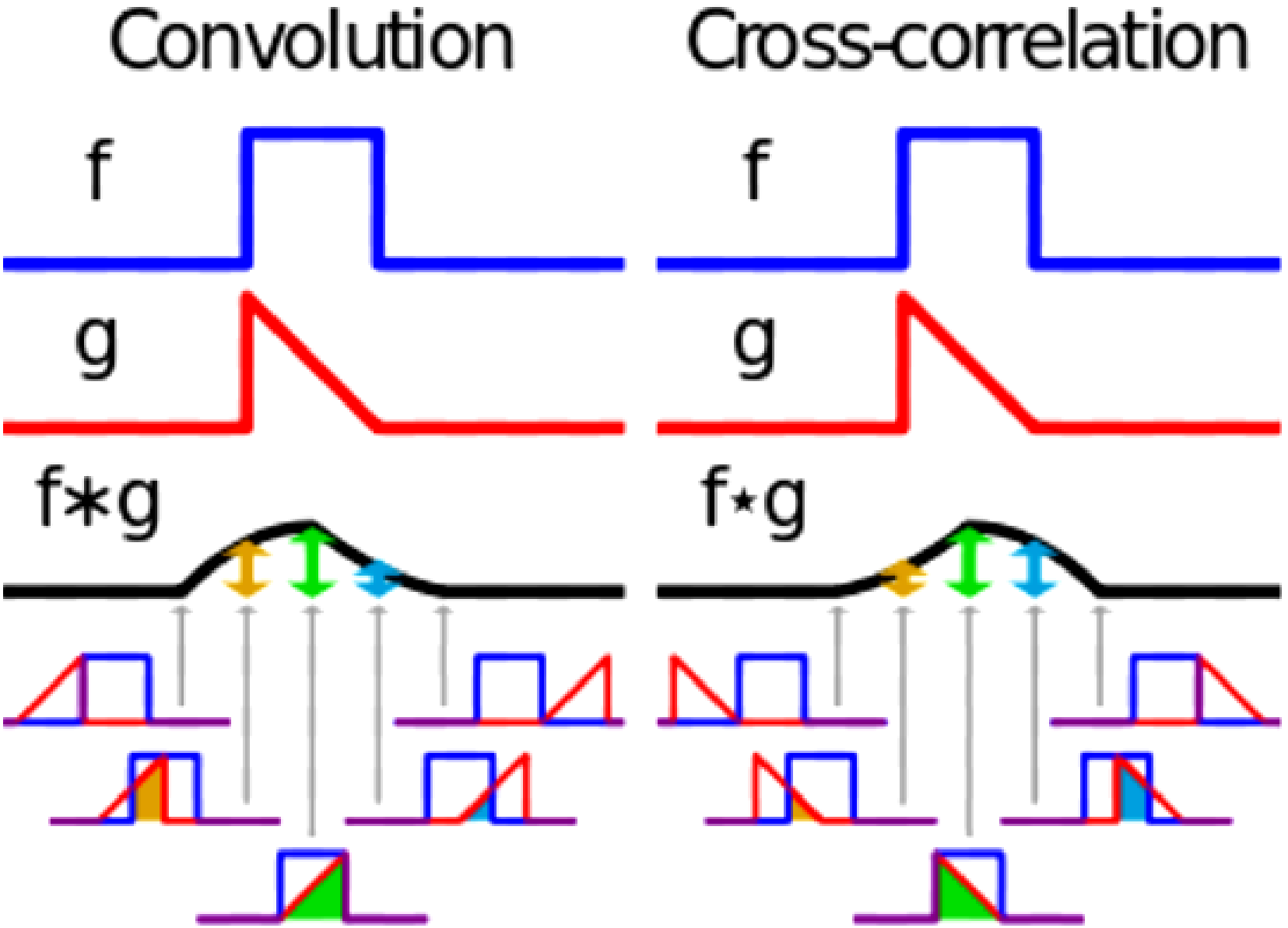
    (3) **Sum** up
  - Cross-correlation **doesn't** involve the **flip** !

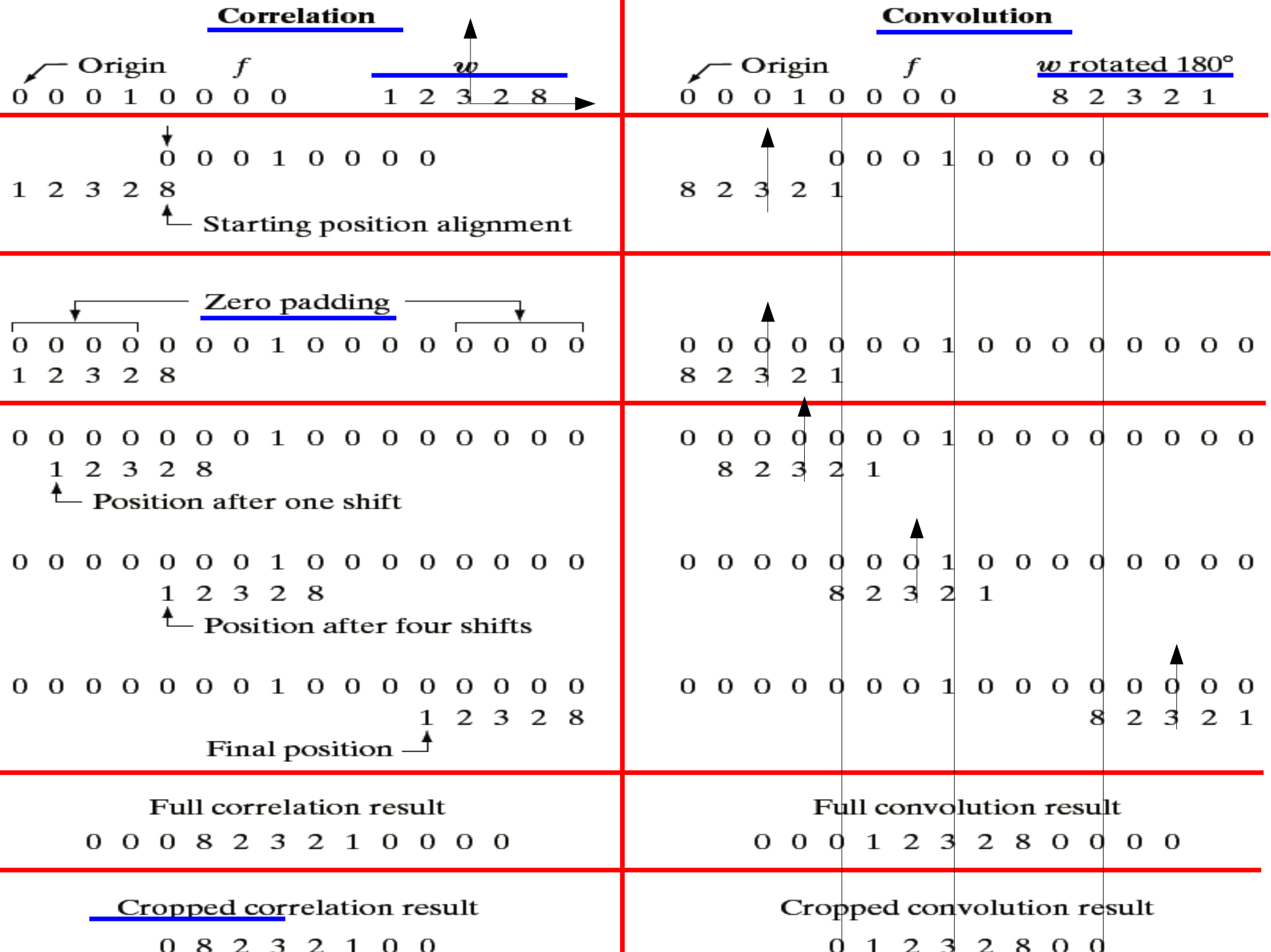
Spatial Filtering

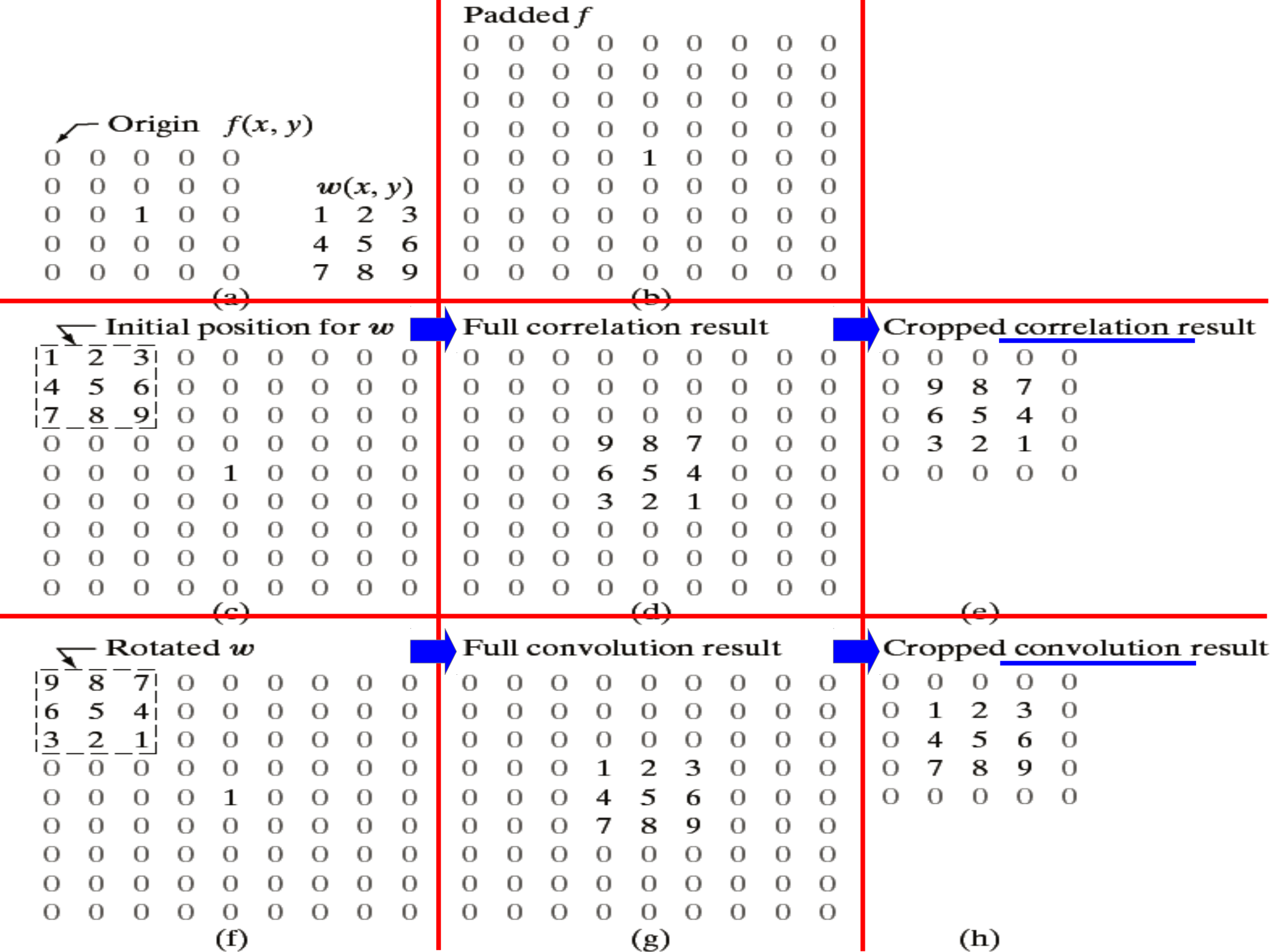
- Cross-Correlation
  - Why does this measure similarity between signals ?
  - Cross-correlation is large when F and shifted-H signals have similar signs
    - When F is positive, shifted-H is also positive
    - When F is negative, shifted-H is also negative
    - i.e., F and shifted-H increase and decrease in sync

Spatial Filtering

- Cross-Correlation




Correlation
Origin
f
w
0 0 0 1 0 0 0 0
1 2 3 2 8
0 0 0 1 0 0 0 0
1 2 3 2 8
Starting position alignment
Zero padding
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after one shift
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after four shifts
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Final position
Full correlation result
0 0 0 8 2 3 2 1 0 0 0 0
Cropped correlation result
0 8 2 3 2 1 0 0
Convolution
Origin
f
w rotated 180°
0 0 0 1 0 0 0 0
8 2 3 2 1
0 0 0 1 0 0 0 0
8 2 3 2 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
Full convolution result
0 0 0 1 2 3 2 8 0 0 0 0
Cropped convolution result
0 1 2 3 2 8 0 0

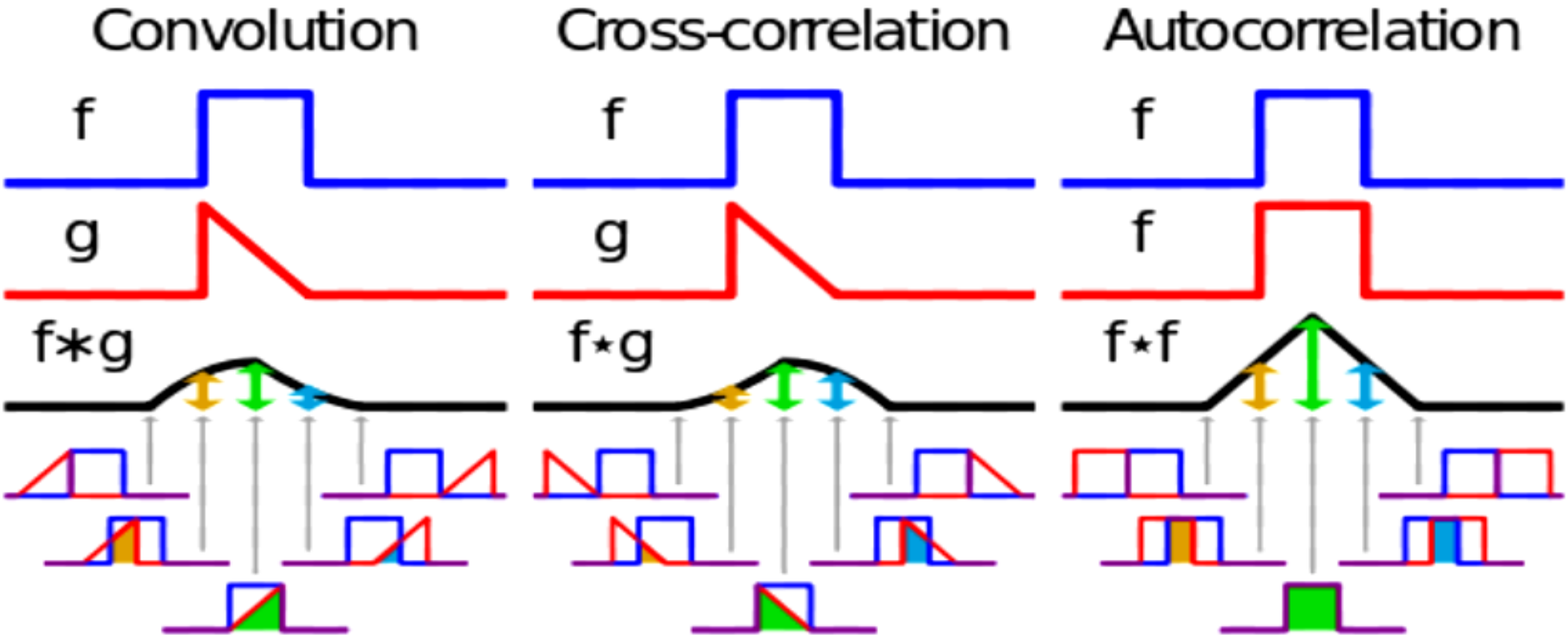

Origin
f(x, y)
0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
w(x, y)
1 2 3
4 5 6
7 8 9
(a)
Padded f
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
(b)
Initial position for w
1 2 3 0 0 0 0 0 0
4 5 6 0 0 0 0 0 0
7 8 9 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
(c)
Full correlation result
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 9 8 7 0 0 0
0 0 0 6 5 4 0 0 0
0 0 0 3 2 1 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
(d)
Cropped correlation result
0 0 0 0 0
0 9 8 7 0
0 6 5 4 0
0 3 2 1 0
0 0 0 0 0
(e)
Rotated w
9 8 7 0 0 0 0 0 0
6 5 4 0 0 0 0 0 0
3 2 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
(f)
Full convolution result
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 1 2 3 0 0 0
0 0 0 4 5 6 0 0 0
0 0 0 7 8 9 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
(g)
Cropped convolution result
0 0 0 0 0
0 1 2 3 0
0 4 5 6 0
0 7 8 9 0
0 0 0 0 0
(h)

Spatial Filtering

- w(.) = convolution mask
- Convolution property
  - If convolution mask is symmetric about x and y axes, then convolved image = cross-correlated image !
  - See previous slide ...

Spatial Filtering

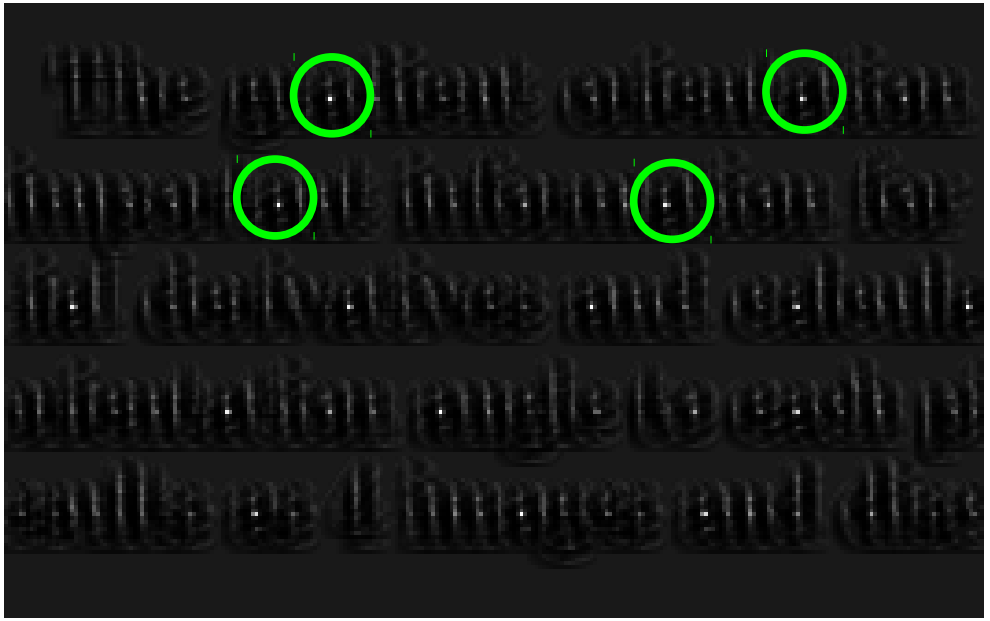
- **Auto-correlation**
  - Cross-correlation of a signal with itself
    - Similar to inner product of signal with itself
    - Reaches a maximum when shift = 0
    - Other shifts may also reach maximum (e.g., periodic signals)

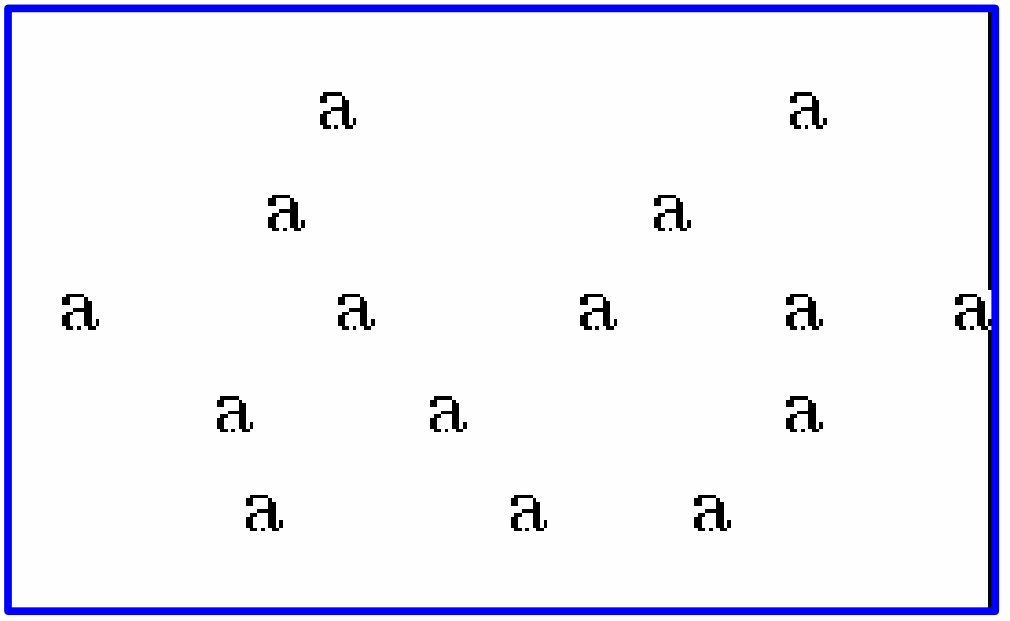


Spatial Filtering

- Cross-correlation applied for object detection
  - Template matching
  - Template = a
    - Binary image {-1, 1}
  - Image = (below)
  - Cross correlation (down)

The gradient orientation
important information for
tial derivatives and calcula
orientation angle to each pi
esults as 4 images and disc

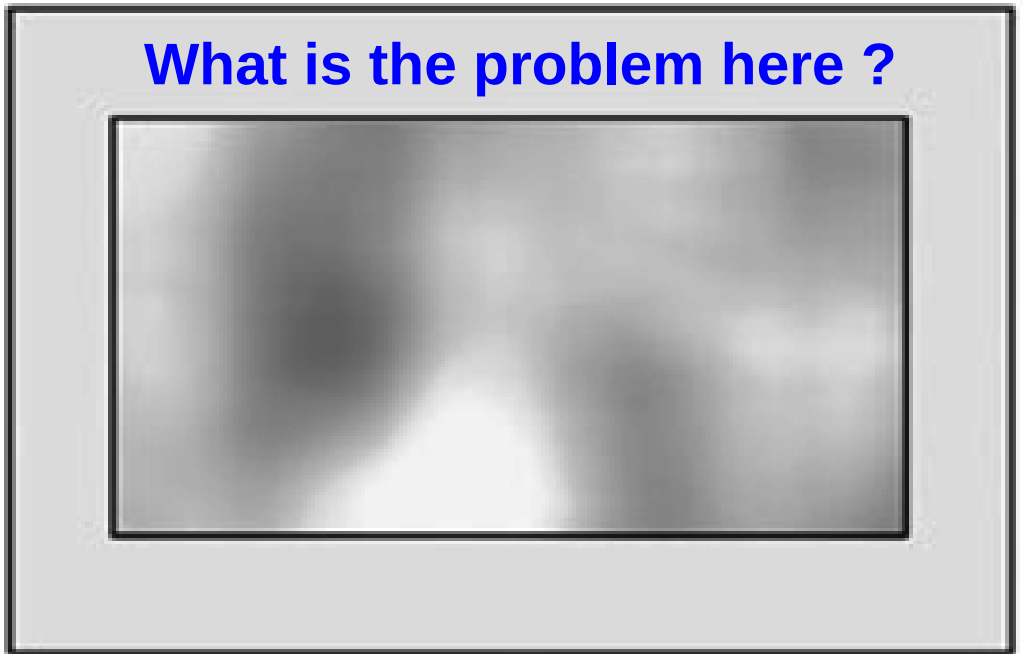




Input image

Template

Correlation



Spatial Filtering

- **Normalized** Cross-Correlation
  - Motivation: Want to measure similarity between 2 signals, without considering effects of :
    - (1) Intensity scaling
    - (2) Intensity shifts
  - How to do this algorithmically ? NCC (F, H) = ?

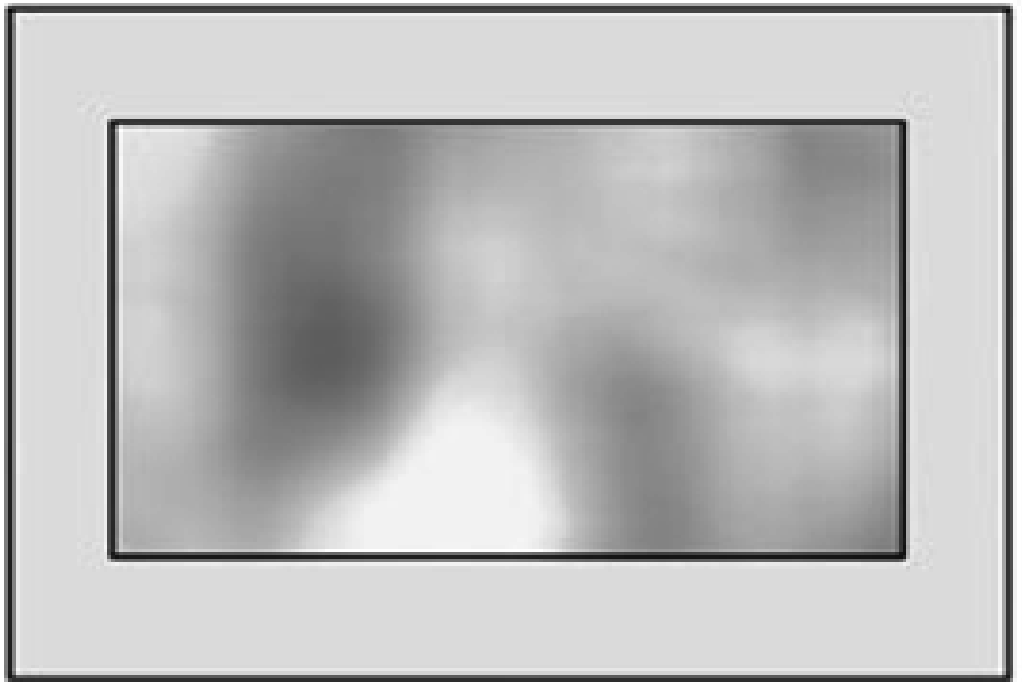
normalization

- (1) Subtract, from F, mean of F → F'
  - Handles changes to intensity shifts
- (2) Divide each value in F' by the root-mean-square of intensities in F' → F''
  - Handles changes in intensity scale
- (3) Do the same for H → H''
- (4) Perform cross-correlation between F'' and H''

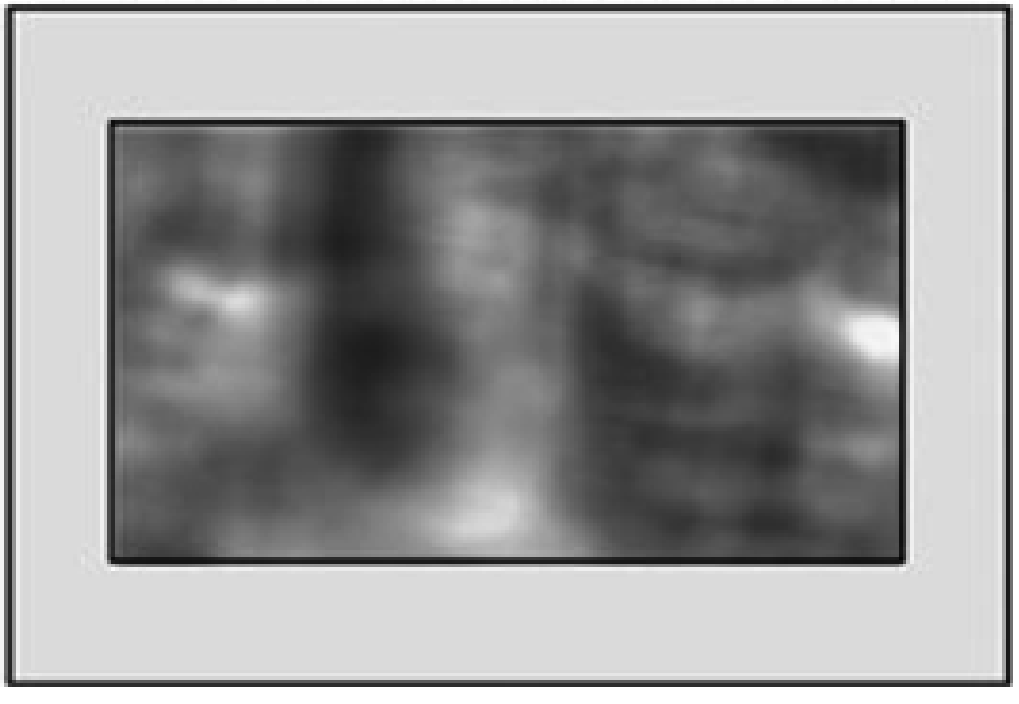
Input image

Template

Correlation



Normalized cross correlation



Spatial Filtering

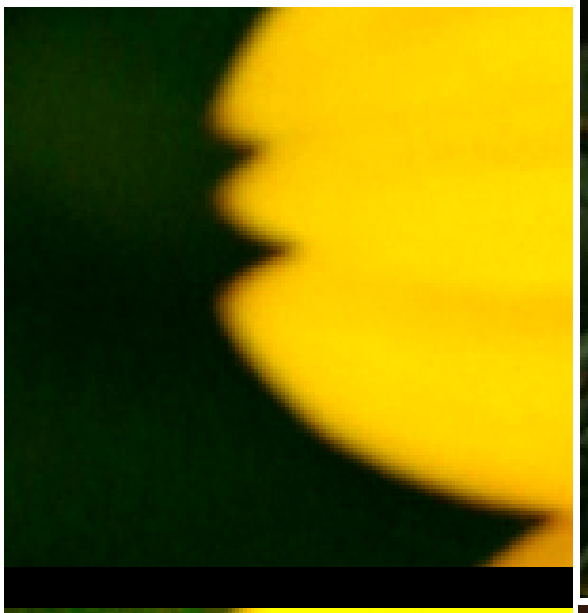
- Applications of spatial filtering
  - Blur / smooth image
  - Reduce image noise
  - Sharpen image
  - Edge detection
    - What is an edge ?
  - ...
  - How do we design weights / masks w(i,j) for these tasks ?

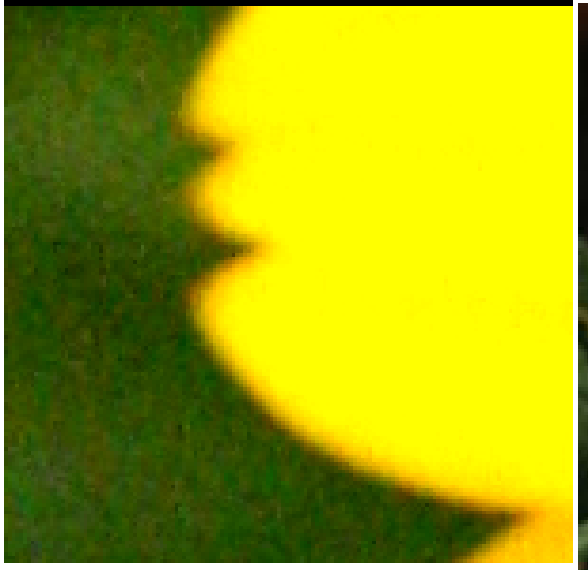
Spatial Filtering

- Image noise
  - Corruption in the image in the form of **random fluctuations** in image intensities
  - Caused by:
    - Measurement errors
    - Fluctuations in signal itself

Spatial Filtering

- Types of image noise
  - Gaussian (additive) →
    - Mean zero
  - Salt and pepper / impulse
    - e.g., intensity spikes
    - e.g., data transmission error





Spatial Filtering

- Increasing levels of Gaussian noise
  - Increasing variance (or standard deviation)



Spatial Filtering

- Increasing levels of salt-and-pepper noise
  - Increasing the number of pixels that get assigned black or white intensities

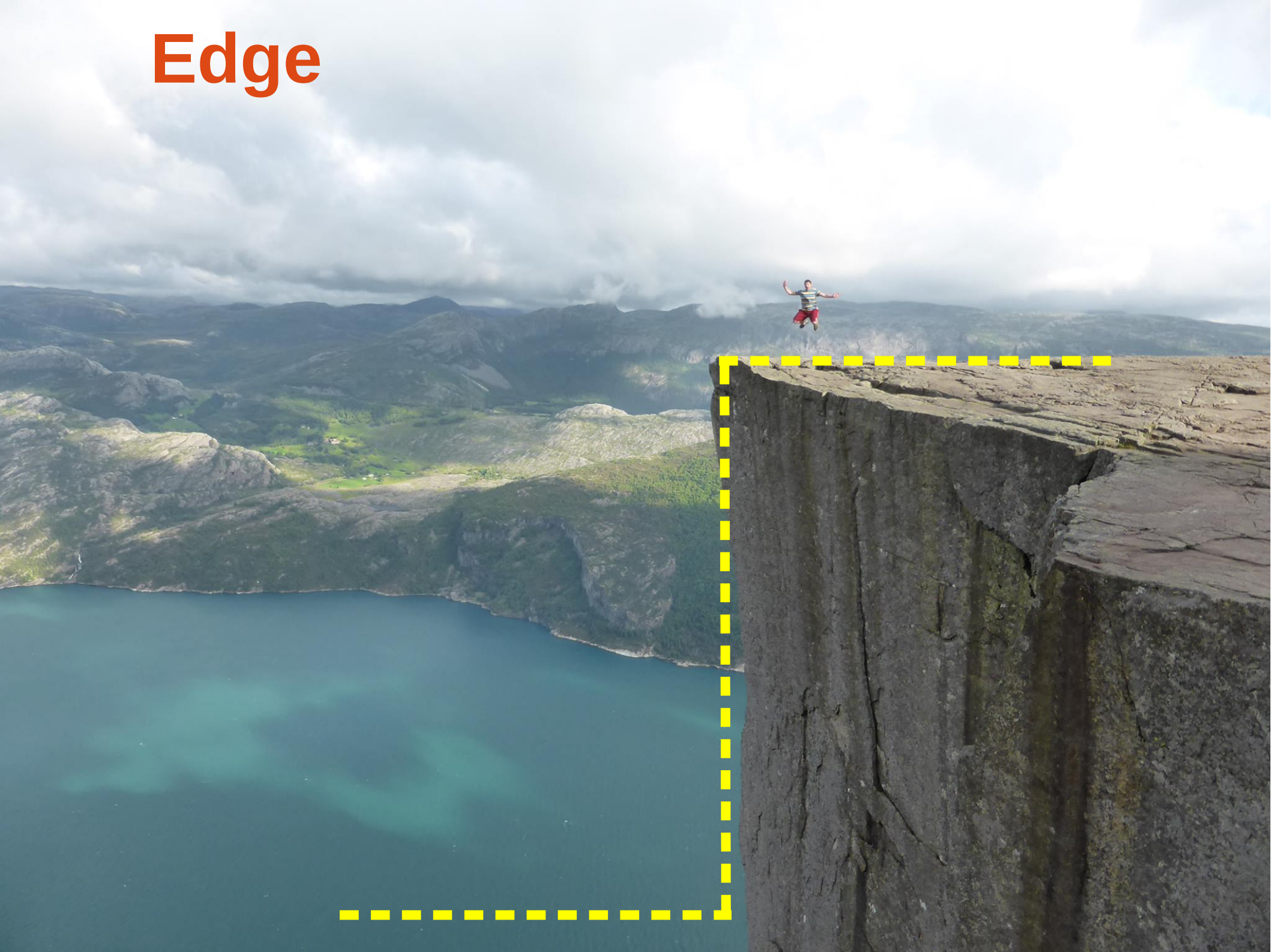
Spatial Filtering

- **Mean filter** (also called **box filter**)
  - Replace pixel value by **mean** of pixel values in window around pixel
  - Mask for convolution
    - We want sum of weights = 1
    - Why ?
      - To preserve intensity range of the original image in the averaged image
    - How to handle boundary pixels ?
      - If mask cropped, then redefine scaling factor
  - How does the mean filter affect edges ?

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Edge

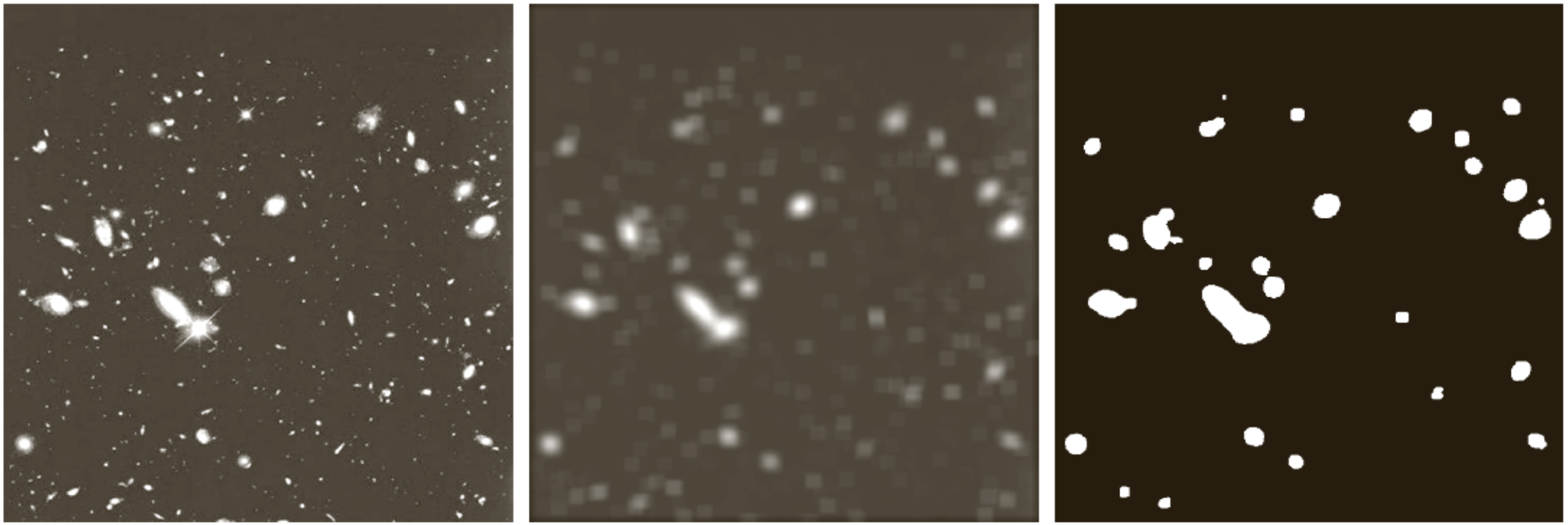


Spatial Filtering

- Mean filter
  - Different sizes
  - Original, box 3
    box 5, box 9,
    box 15, box 25
  - Circle diameter
    (in image)
    = 25 pixels

Spatial Filtering

- Mean filter
  - Left: image from Hubble space telescope
  - Middle: mean-filtered image
  - Right: thresholded image
  - Notice the **square-shaped** blobs (artifact)



Spatial Filtering

- How fast is convolution ?
  - If image size = M x N
  - If mask size = P x Q
  - Then, convolution takes M N P Q operations
  - Can this be made faster ?
    - In some special cases ?
    - Yes ! How ?

Spatial Filtering

- How fast is convolution ?
  - Observe a trick
    - Let U = 1D column image = [u1 u2 u3 u4]^T
    - Let V = 1D row image = [v1 v2 v3]
    - Then, 2D convolution of U and V = ?
    - = (outer) product of U and V

U * V = U V = $$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 \end{bmatrix}$$

Spatial Filtering

- How fast is convolution ?
  - Combine ideas from (1) trick (2) associativity of convolution
  - Let F be a 2D image
  - Let U = 1D column image
  - Let V = 1D row image
  - Then, (U * V) * F = U * (V * F)
    - U * V is a 2D mask
    - V * F = convolution of each row of F with V
      - Output F1 = same size (or slightly larger) than F
    - U * F1 = convolution of each column of F1 with U
      - Output F2 = same size (or slightly larger) than F1

Spatial Filtering

- How fast is convolution ?
  - Let F = image of size M x N
  - Let H = convolution mask of size P x Q
  - **If H = outer product U * V**
    - U is of size P x 1
    - V is of size 1 x Q
  - **Then, H * F = (U * V) * F = U * (V * F)**
  - This can be computed faster than before:
    - Operations needed for F1 = V * F = Q (M N)
    - Operations needed for U * F1 = P (M N)
    - Total operations needed for U * (V * F) = **M N (P + Q)**

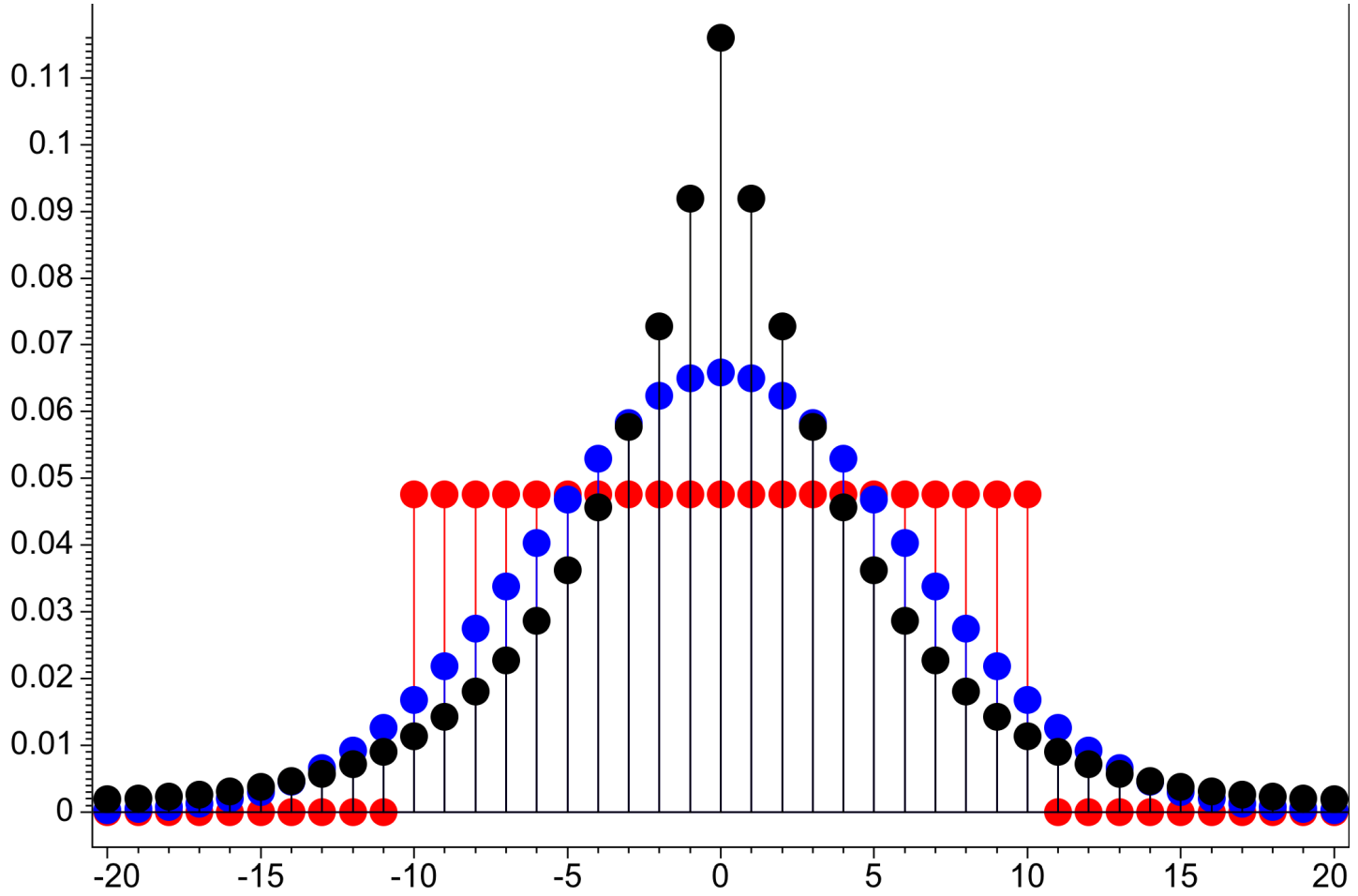
Spatial Filtering

- **Separable** filter = filter that can be written as an outer product
  - Separable filter leads to faster algorithm for convolution
  - Is the mean filter separable ?

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

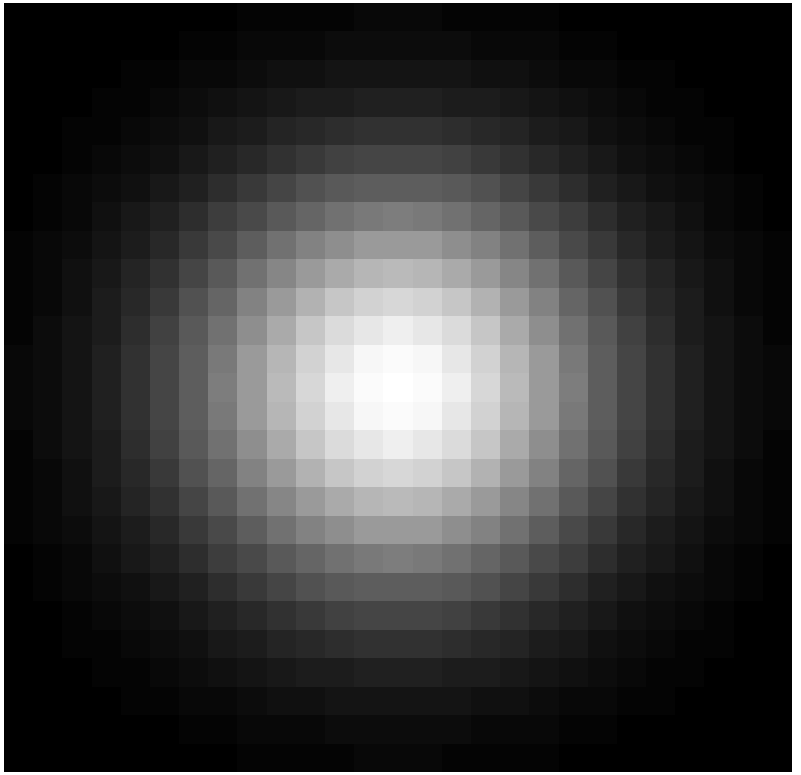
Spatial Filtering

- Weighted-mean filter
  - Mean filter = red
  - Gaussian-weighted filter = blue, black
    - Assign more weight to intensities at pixels closer to center pixel



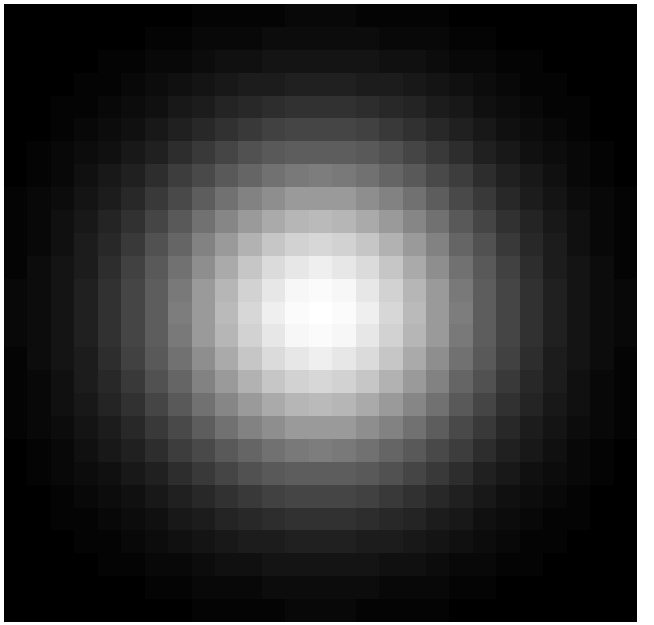
Spatial Filtering

- Weighted-mean filter
  - 2D symmetric Gaussian mask (before rescaling)
    - Standard deviation = sigma = 2
    - $G(x,y) = \exp(-0.5 (x^2 + y^2) / \sigma^2)$
      - Ignore density normalization factor
    - Is this filter separable ?



Spatial Filtering

- Weighted-mean filter
  - 2D symmetric Gaussian mask (before rescaling)
    - Standard deviation = sigma = 2

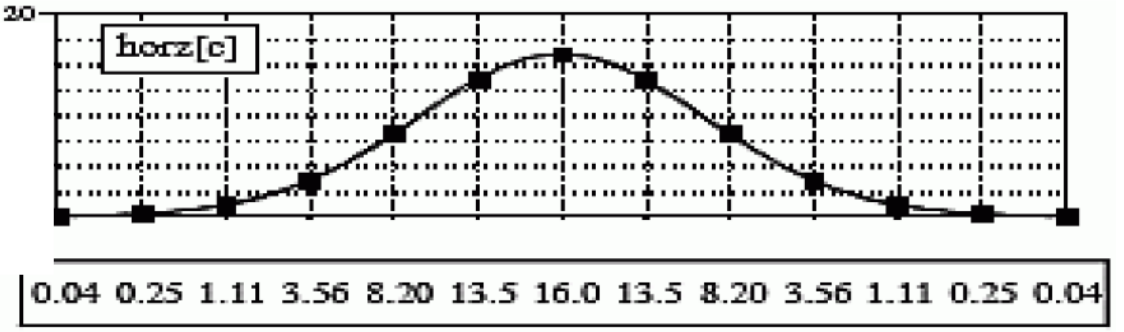


$$g(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

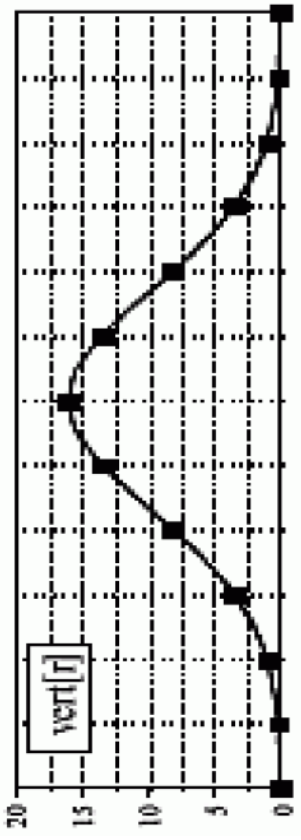
0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	3	11	26	50	73	82	73	50	26	11	3	0
1	6	20	50	93	136	154	136	93	50	20	6	1
2	9	30	73	136	198	225	198	136	73	30	9	2
2	11	34	82	154	225	255	225	154	82	34	11	2
2	9	30	73	136	198	225	198	136	73	30	9	2
1	6	20	50	93	136	154	136	93	50	20	6	1
0	3	11	26	50	73	82	73	50	26	11	3	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0

Spatial Filtering

- Gaussian filter is separable
  - When covariance matrix is diagonal



0.04	0.25	1.11	3.56	8.20	13.5	16.0	13.5	8.20	3.56	1.11	0.25	0.04



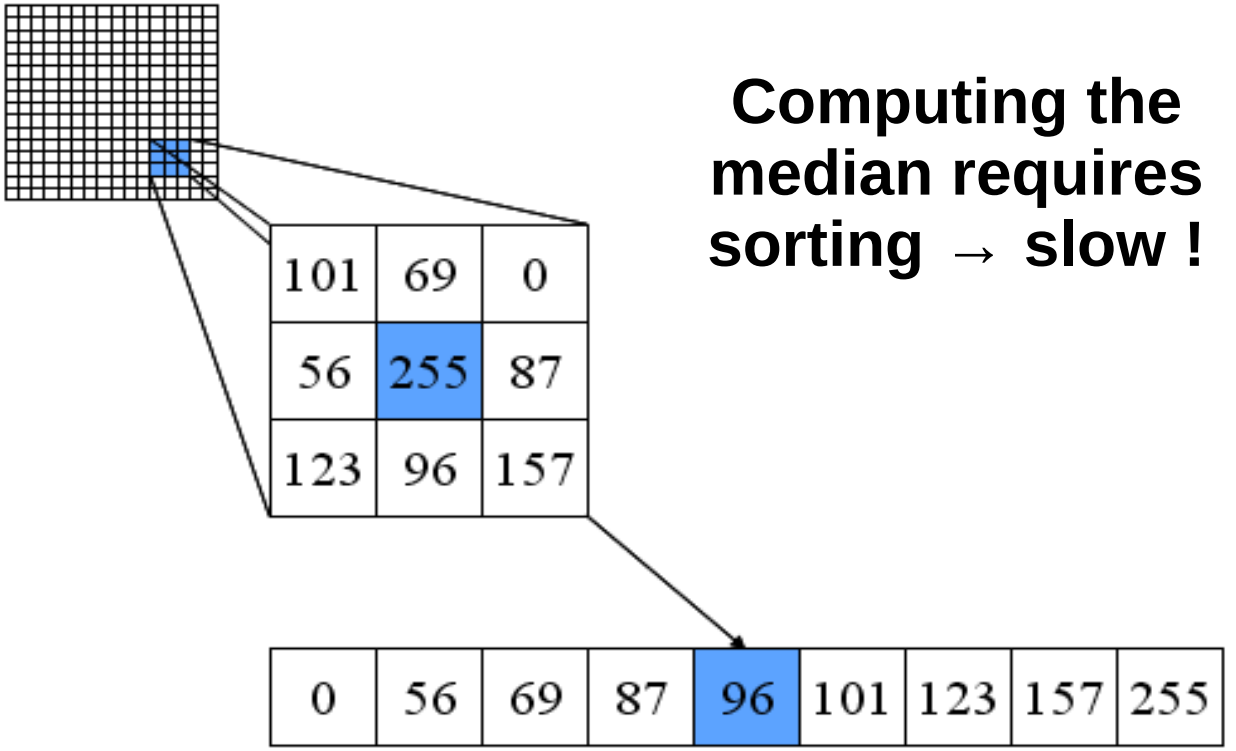
0.04	0	0	0	0	0	1	1	1	0	0	0	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
16.0	1	4	18	57	131	216	255	216	131	57	18	4	1
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
0.04	0	0	0	0	0	1	1	1	0	0	0	0	0

Spatial Filtering

- Weighted-mean filter versus mean filter
  - Trade offs
  - Assuming mask size is same, ...
  - Weighted-mean filter:
    - (1) blurs less and preserves more edges
    - (2) removes less noise in constant regions
    - (3) rotation-invariant filtering

Spatial Filtering

- Median filter
  - Replace pixel value by **median** of pixel values in window around pixel
    - For P(X), median = m, s.t. P (X >= m) >= 0.5, P (X <= m) >= 0.5



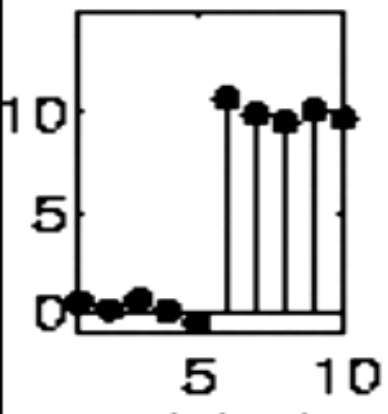
Computing the median requires sorting → slow !

  - Trick question
    - How can the median filter be described as a convolution ?

Spatial Filtering

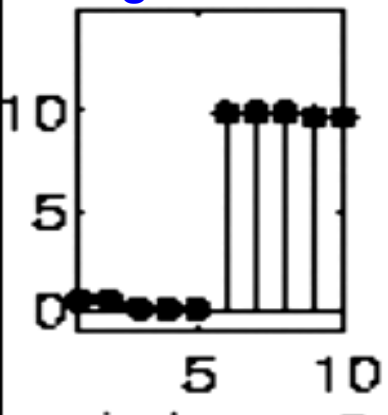
- Median filter
  - Better for removing salt-and-pepper noise, as compared to mean filter
  - Why ?
    - Because (a small level of) salt-and-pepper noise **doesn't** affect the **median** of the original intensities
    - Because (any level of) salt-and-pepper noise **does** affect the **mean** of the original intensities
    - Median is **robust** to **outliers.** Mean isn't.
      - **Outlier** = an observation / outcome that is **extremely** different from others in population
  - How does median filter affect an edge within an image ?

MEDIAN

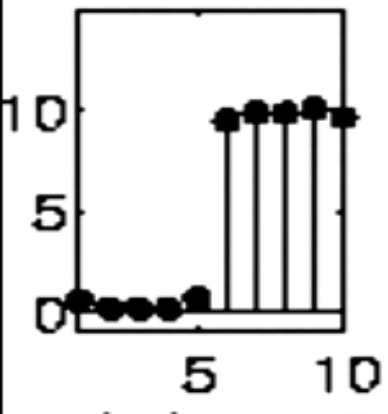


original

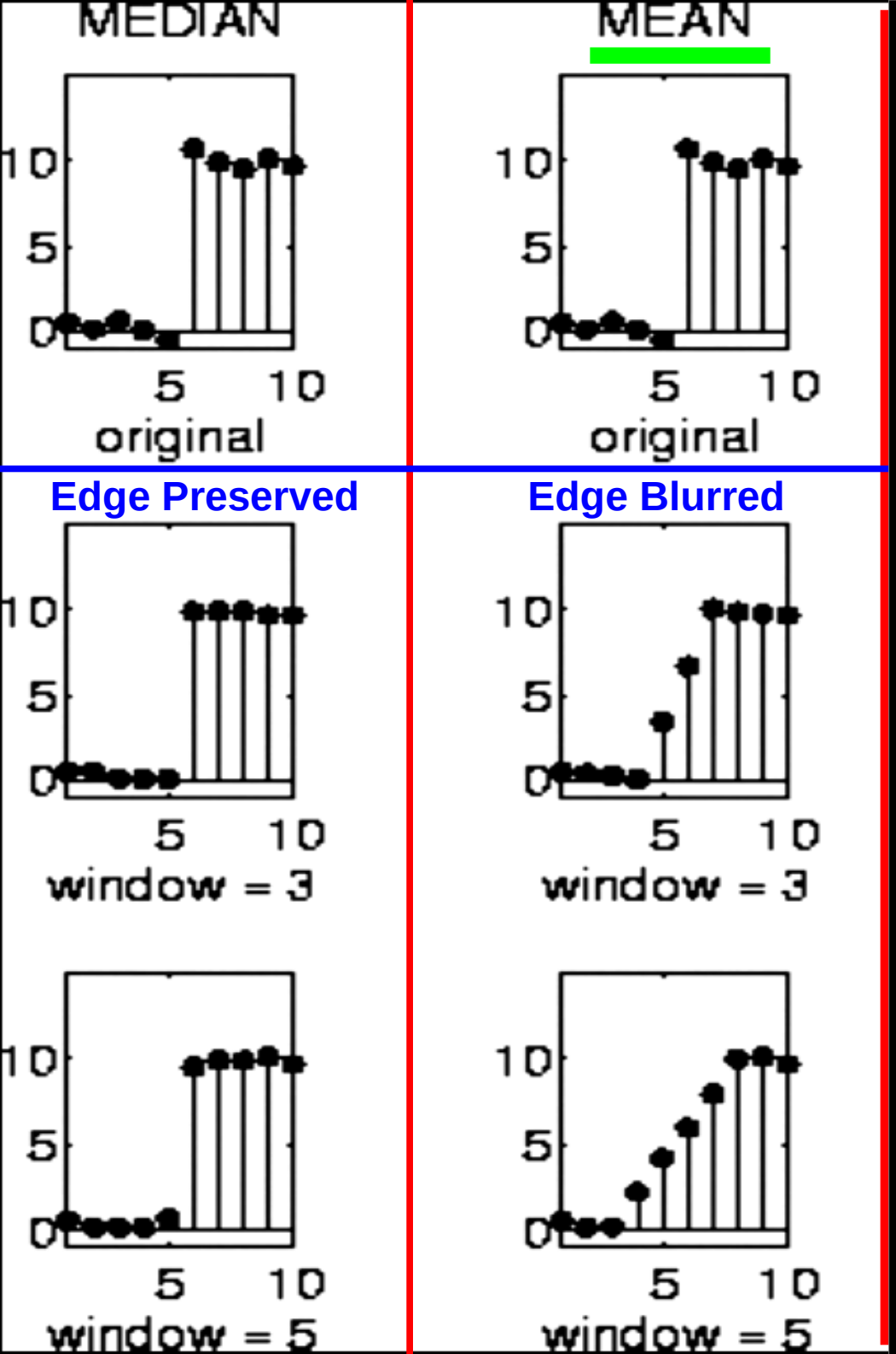
Edge Preserved

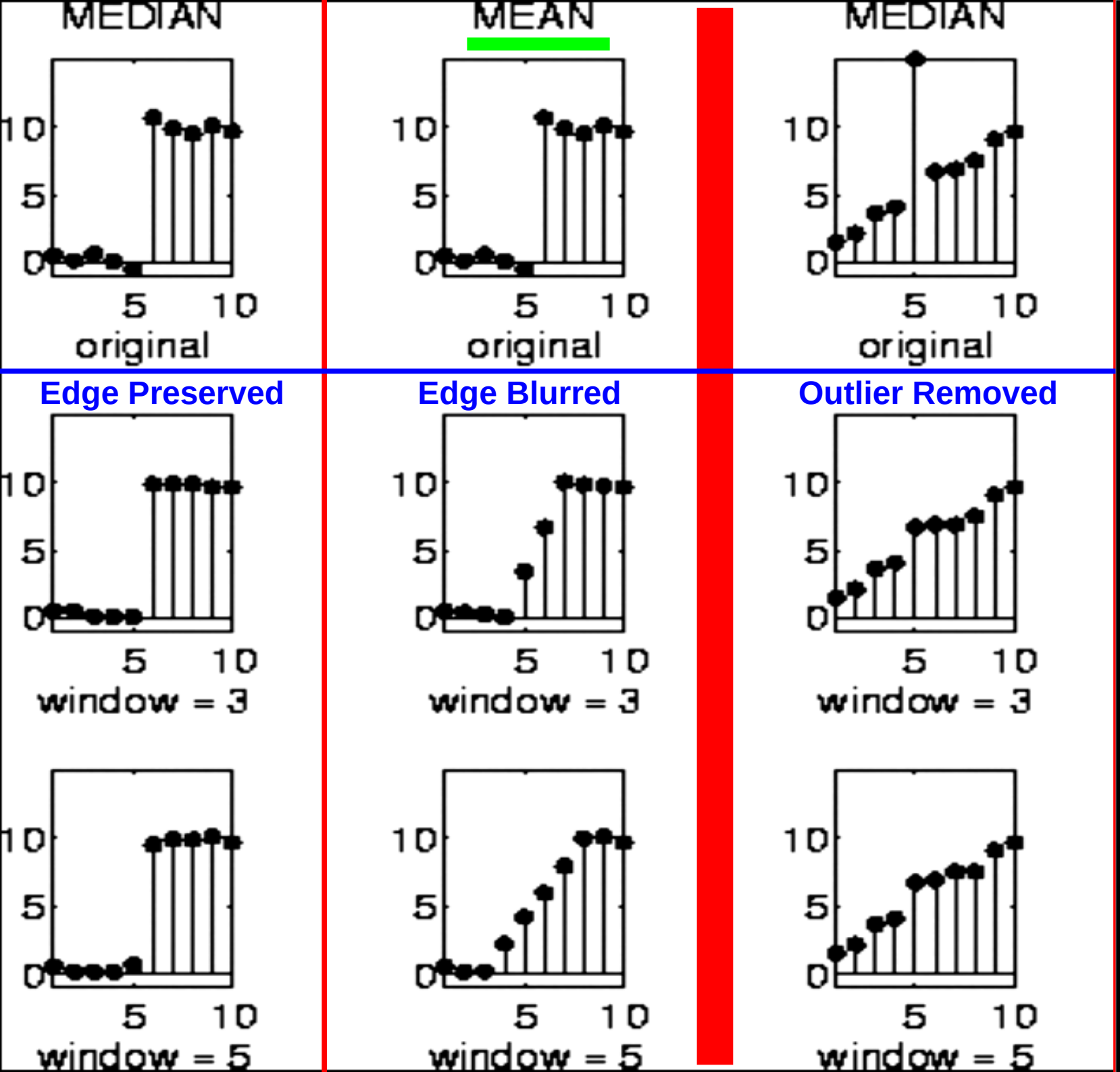


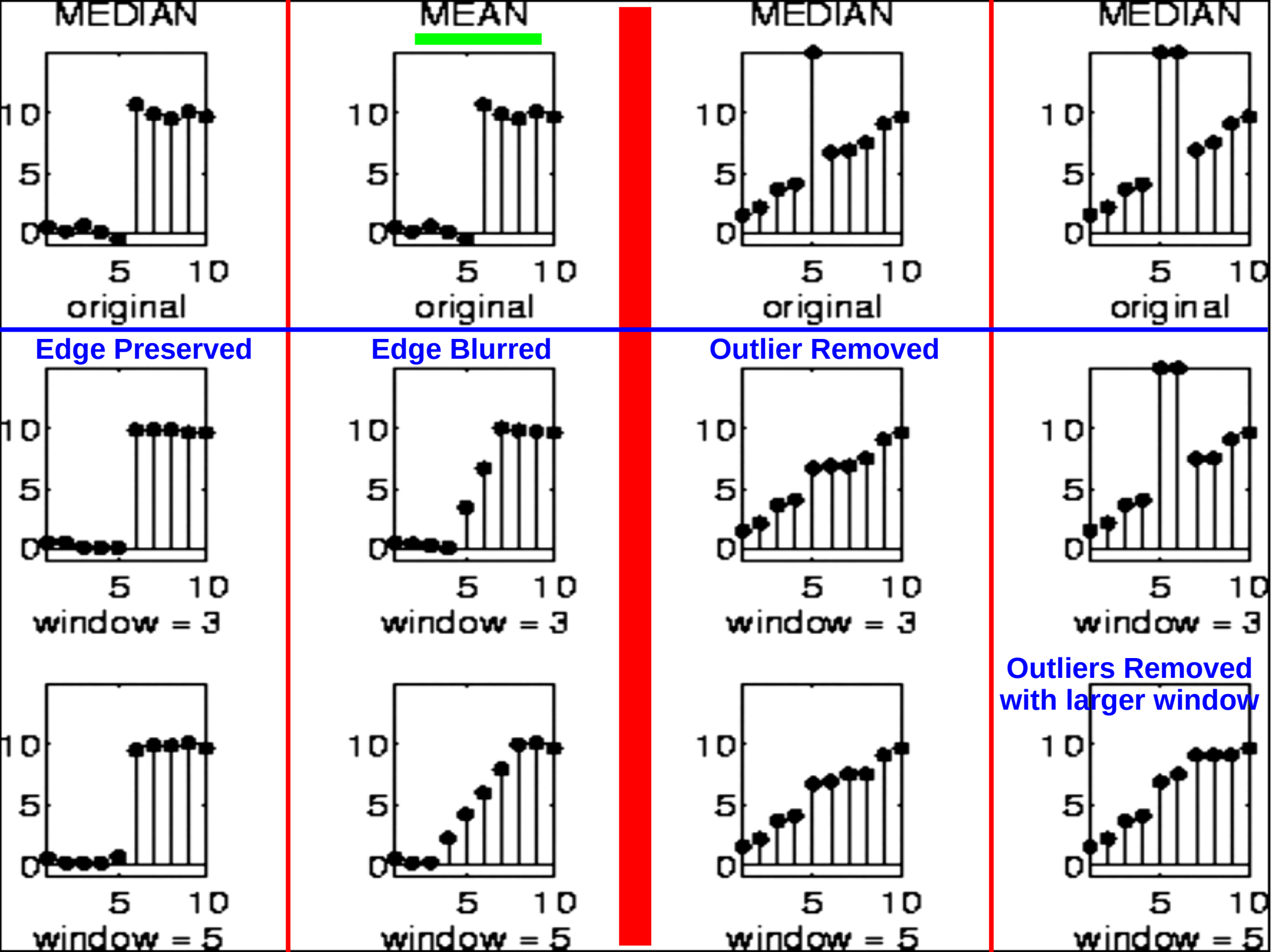
window = 3



window = 5


MEDIAN
10
5
0
5
10
original
MEAN
10
5
0
5
10
original
Edge Preserved
10
5
0
5
10
window = 3
Edge Blurred
10
5
0
5
10
window = 3
10
5
0
5
10
window = 5
10
5
0
5
10
window = 5


MEDIAN
MEAN
MEDIAN
10
5
0
5 10
original
original
original
Edge Preserved
Edge Blurred
Outlier Removed
window = 3
window = 3
window = 3
window = 5
window = 5
window = 5


MEDIAN
MEAN
MEDIAN
MEDIAN
10
5
0
5 10
original
original
original
original
Edge Preserved
Edge Blurred
Outlier Removed
window = 3
window = 3
window = 3
window = 3
Outliers Removed
with larger window
window = 5
window = 5
window = 5
window = 5

Spatial Filtering

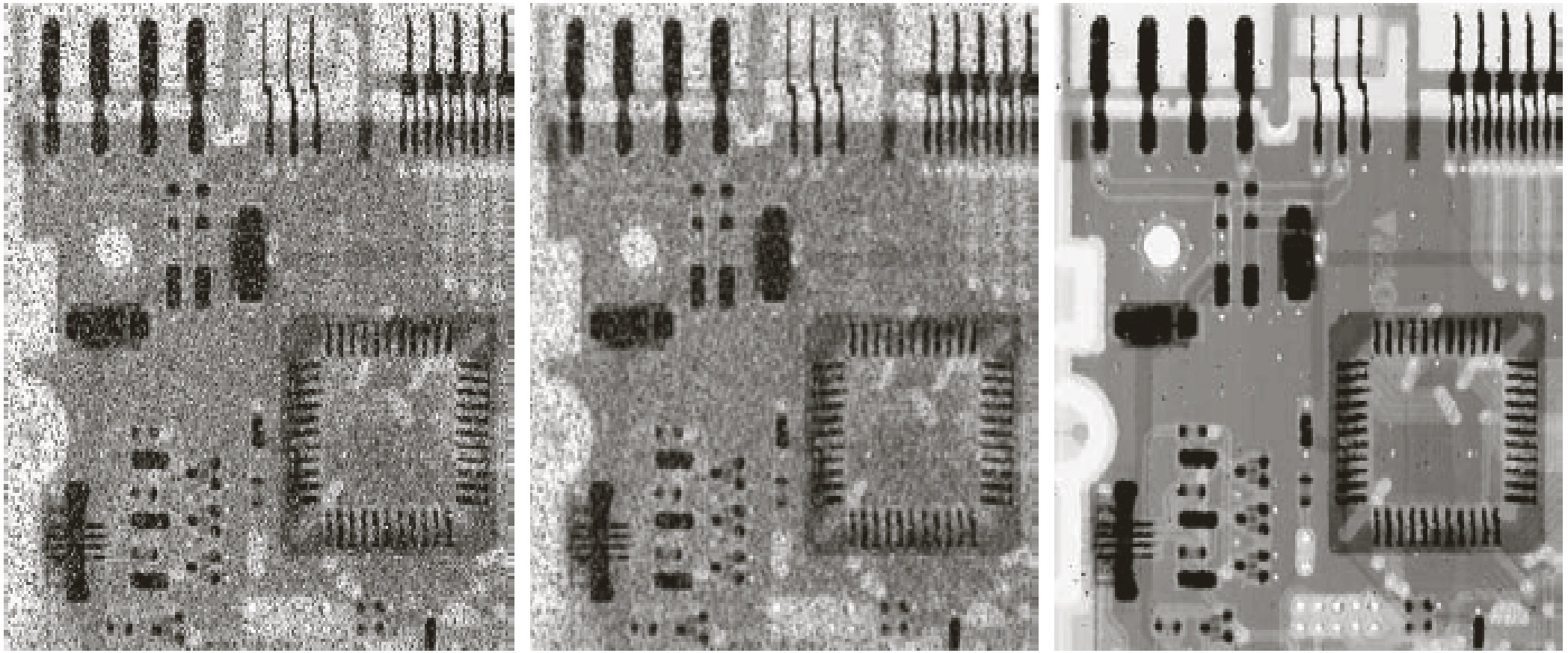
- Median filter on salt-and-pepper noise

Spatial Filtering

- Median filter on salt-and-pepper noise
  - (Surprisingly) Good on large amounts of impulse noise

Spatial Filtering

- Median filter on salt-and-pepper noise



a b c

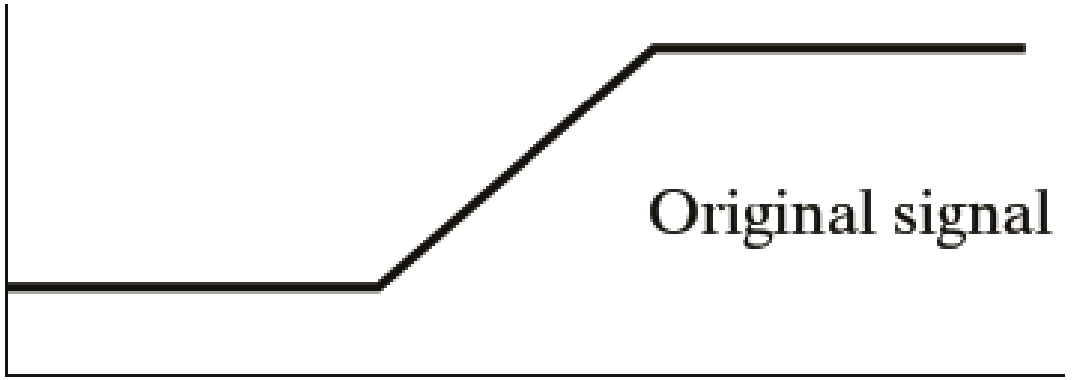
FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

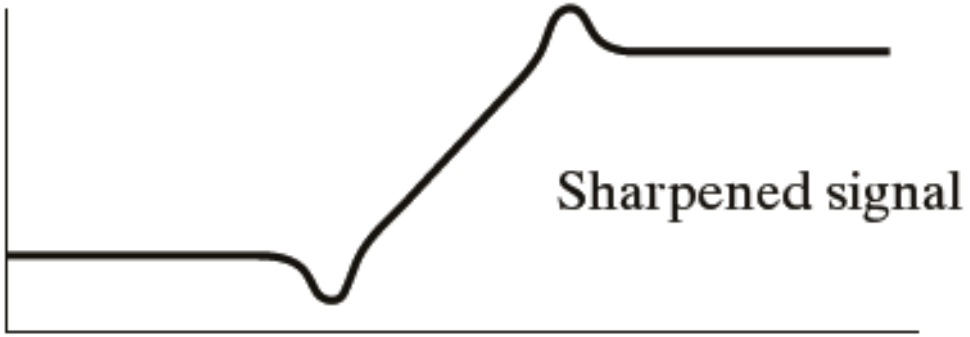
Spatial Filtering

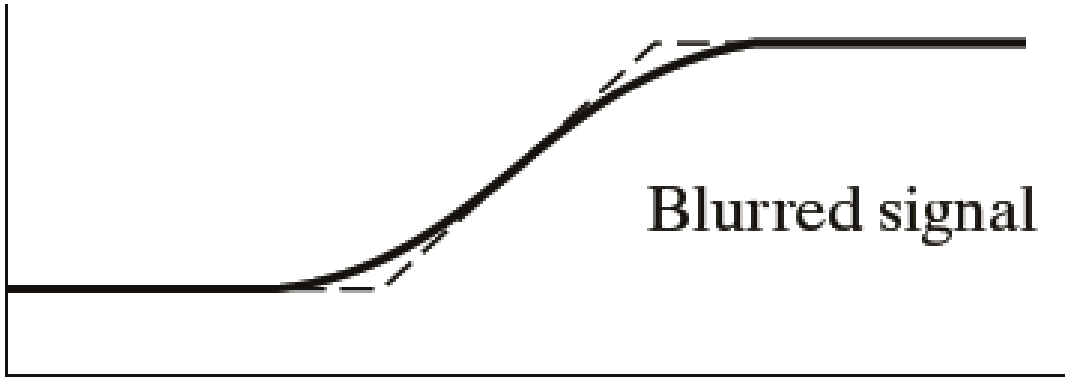
- Median filter on Gaussian noise
  - Can preserves edges better
  - But, can create artificial edges in smooth regions (e.g., forehead)

Spatial Filtering

- Sharpening filter
  - “Opposite” of smoothing or blurring
  - Smoothing performs integration
  - Sharpening should do the opposite !







Spatial Filtering

- Sharpening filter examples

Spatial Filtering

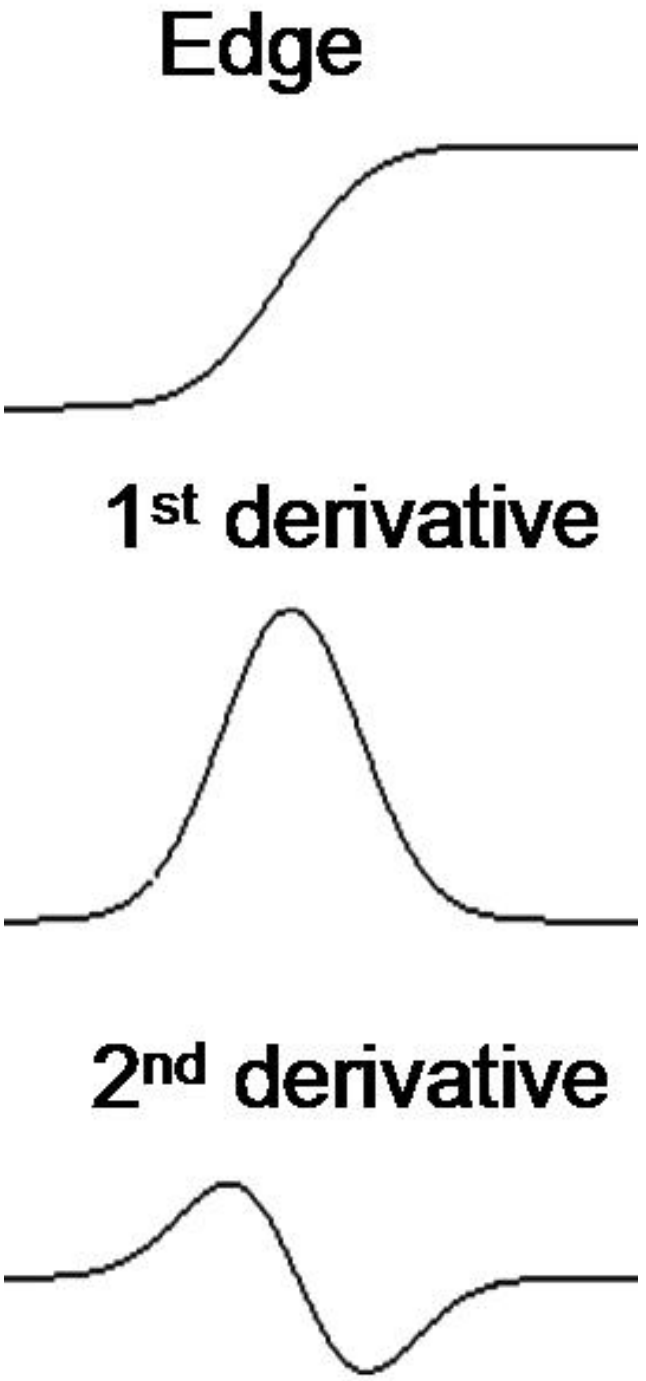
- Image derivatives
  - 1st derivative across edge
    - Gradient in 2D
    - For f (x1, x2, …, xn)

$$\nabla f = \frac{\partial f}{\partial x_1}\mathbf{e}_1 + \cdots + \frac{\partial f}{\partial x_n}\mathbf{e}_n$$

where e1,...,en are basis vectors

  - 2nd derivative across edge
    - Laplacian in 2D
    - For f (x1, x2, …, xn)

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

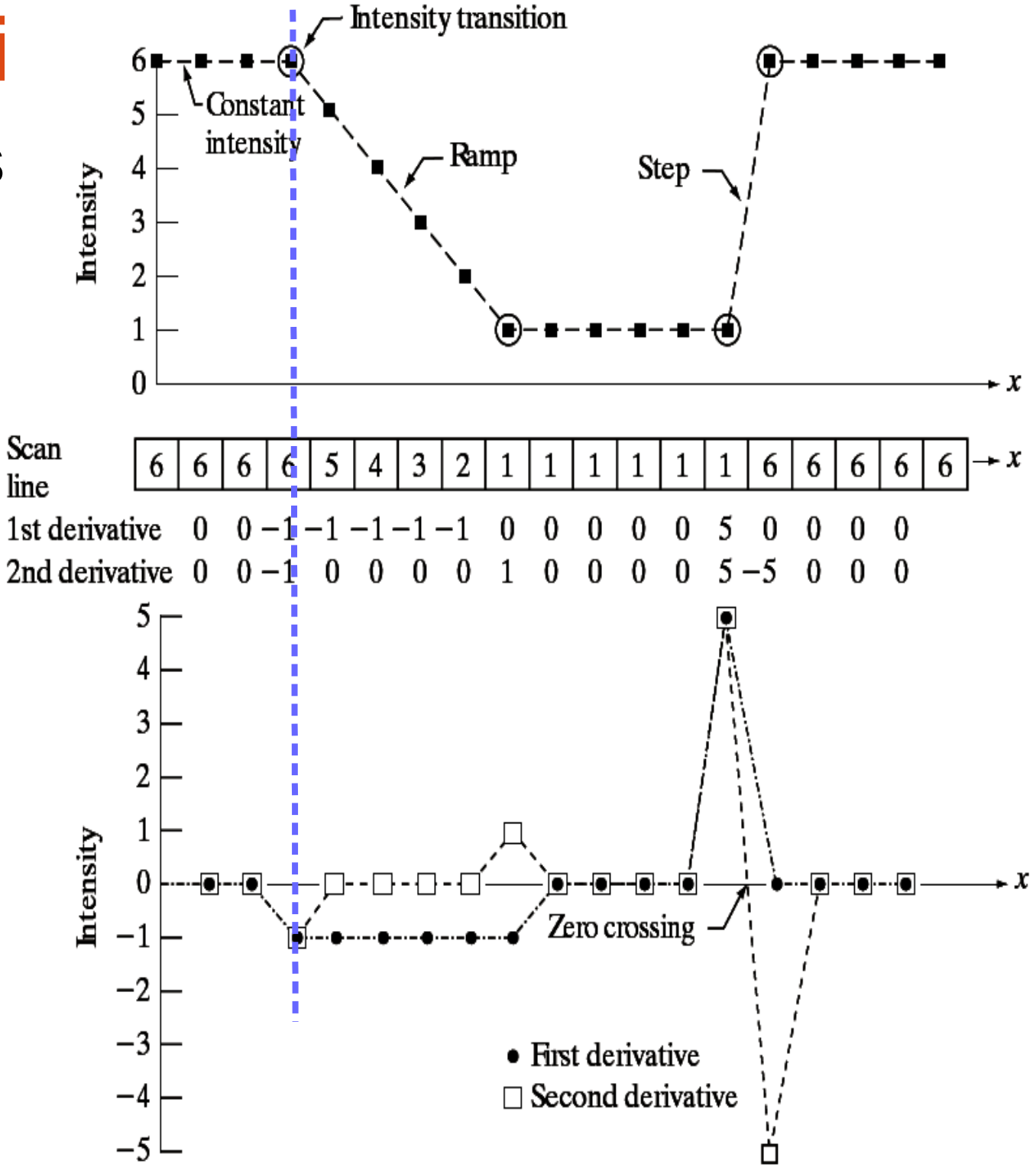


Spatial Filteri

- Image derivatives
  - Very sensitive to noise
    - Example:

      1-sided finite differences

  - "Zero crossing"



Spatial Filtering

- Image derivatives
  - Convolution mask for 1st derivative
    - Use standard 2-sided finite differences
    - Sobel operator
      - Derivative along X axis

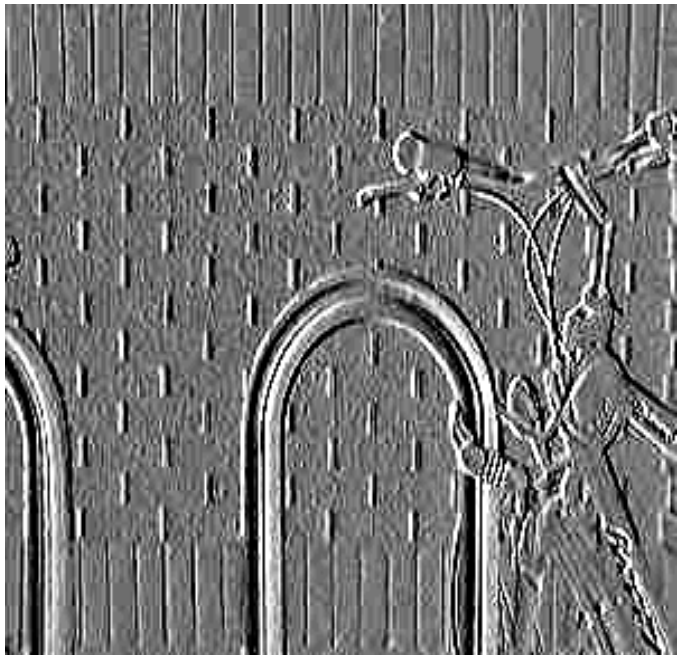
$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

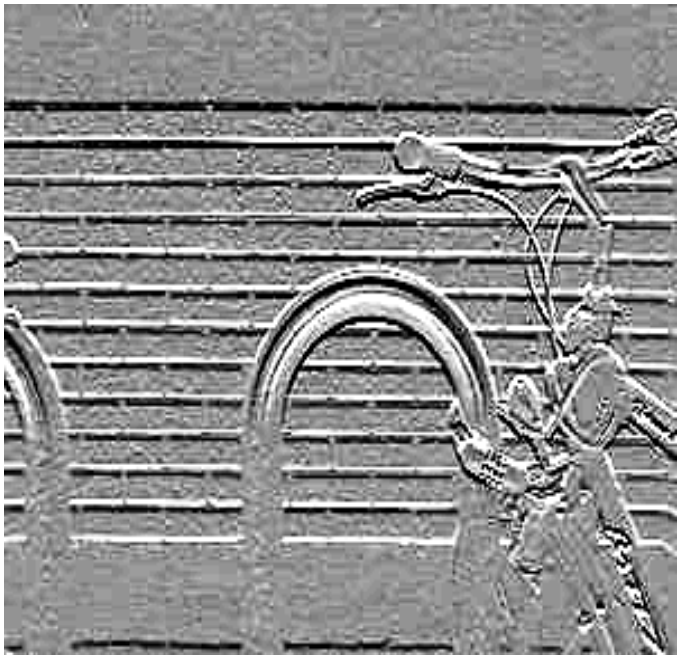
      - Averages derivatives in neighborhood to reduce effect of noise
      - Derivative along Y axis

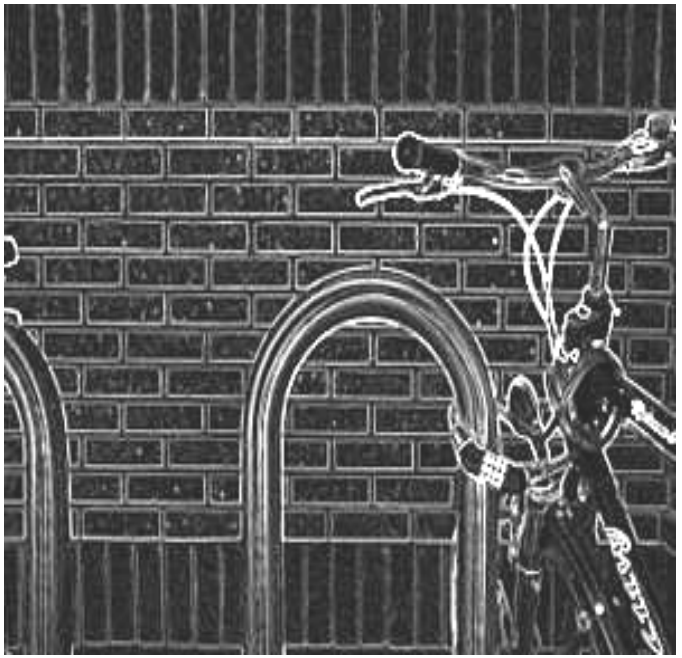
$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Spatial Filtering

- Image derivatives
  - Convolution mask for 1st derivative
    - Sobel operator
      - Averages derivatives in neighborhood to reduce effect of noise

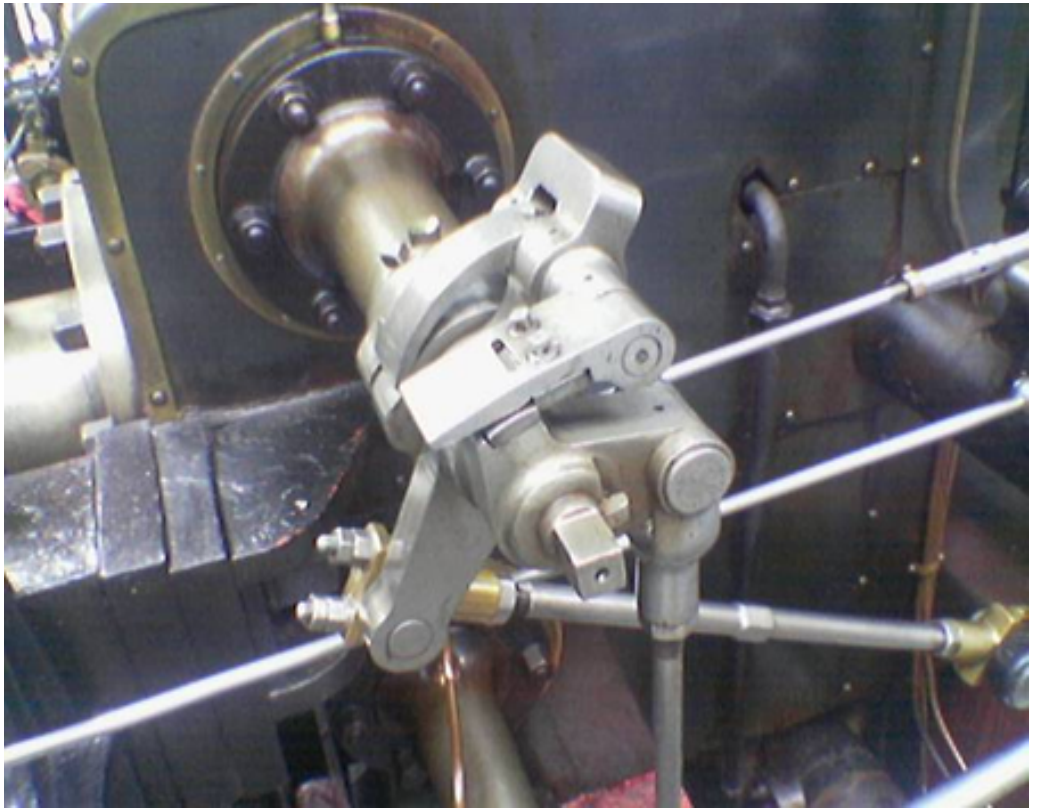


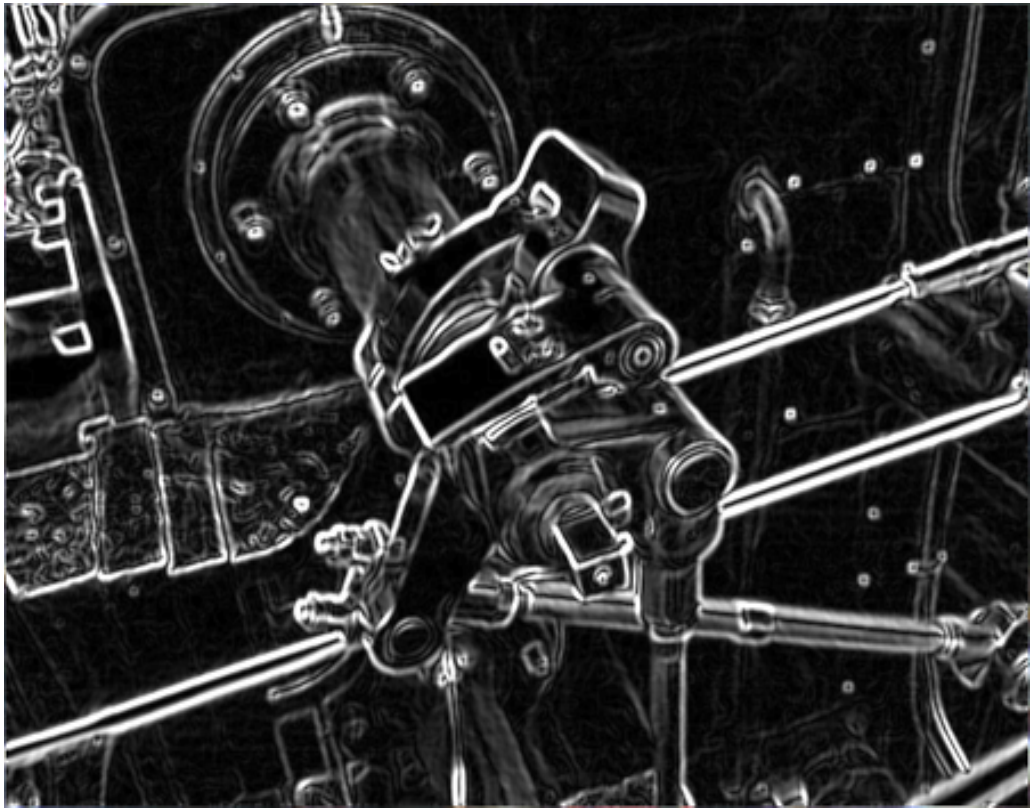




Spatial Filtering

- Image derivatives
  - Convolution mask for 1st derivative
    - Sobel operator
      - How to apply it on a RGB color image ?





Spatial Filtering

- Image derivatives
  - Convolution mask for second derivative
    - Use standard finite differences
    - In 1D : $\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$
    - In 2D
      - Laplacian:
        Add 2nd derivatives along X, Y directions

0	1	0
1	−4	1
0	1	0

      - Add 2nd derivatives along X, Y, diagonal directions

1	1	1
1	−8	1
1	1	1

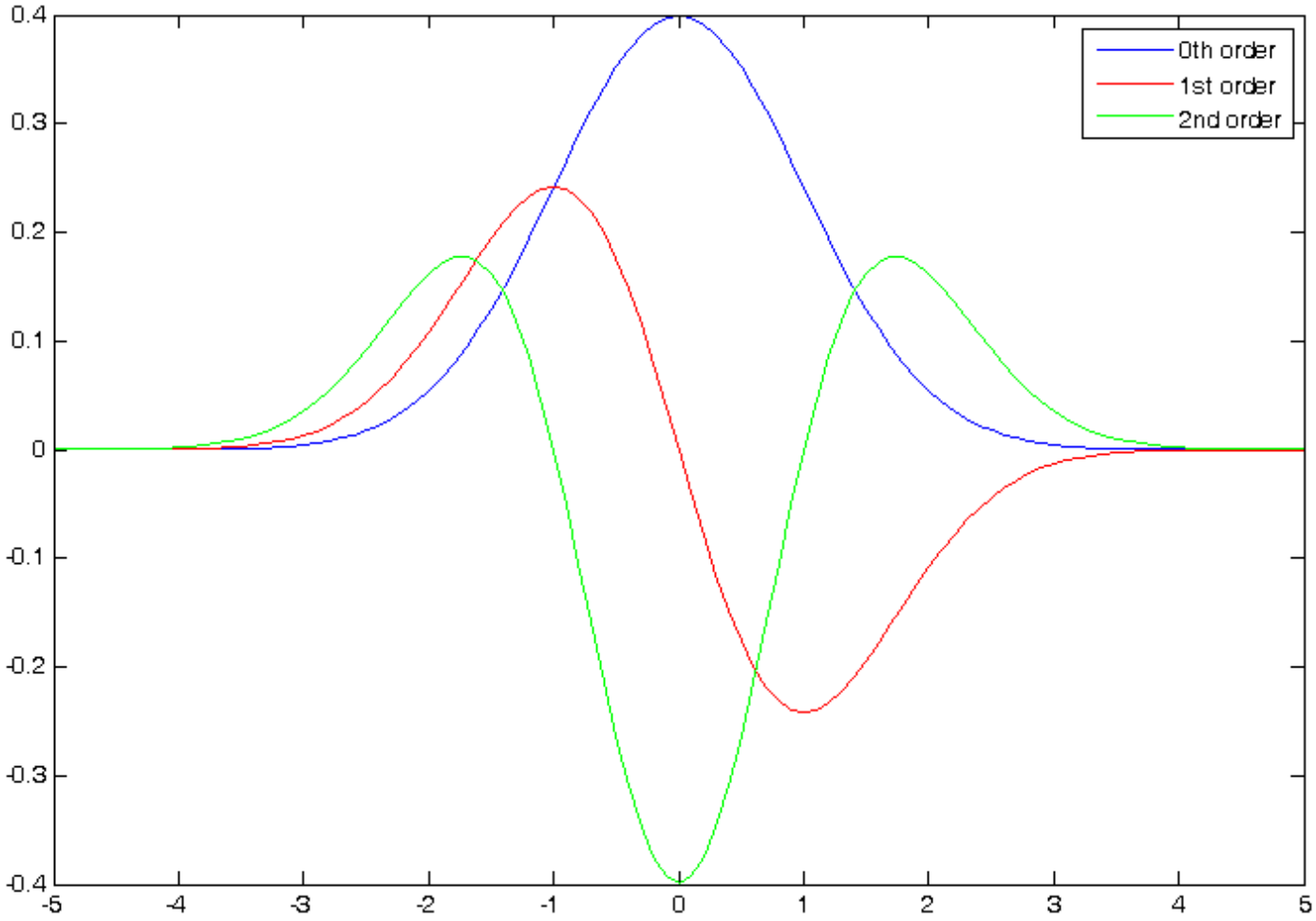
    - In practice, this isn't preferred.
      See next slide for what is done.

Spatial Filtering

- Image derivatives
  - **Laplacian of Gaussian (LoG)**

    (1) Apply a Gaussian filter to smooth image (reduce noise)

    (2) Apply the Laplacian filter to the smoothed image

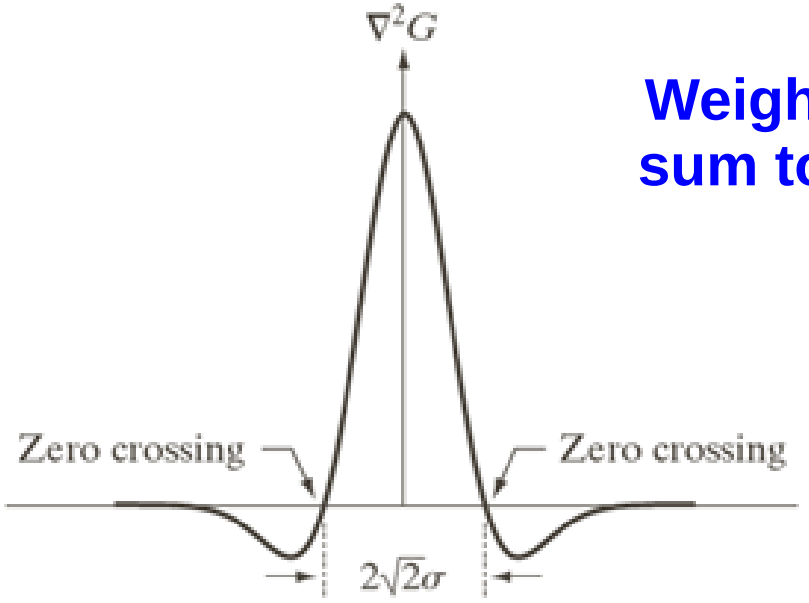


$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Negative of LoG

a b
c d

FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

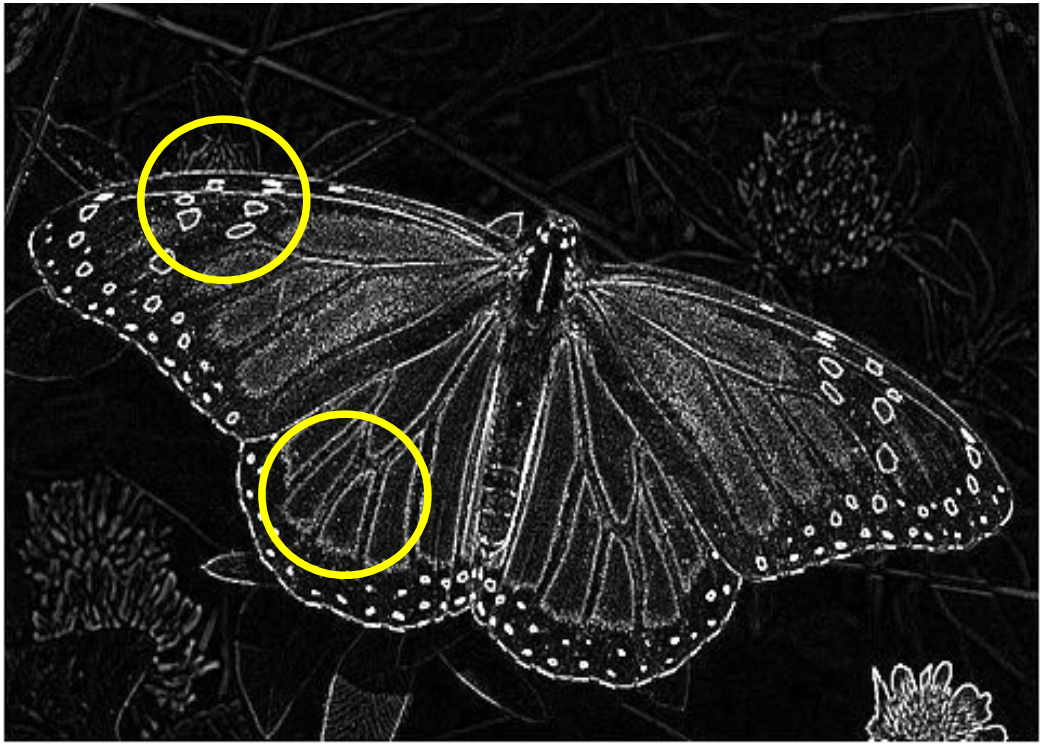


Weights sum to 0

0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

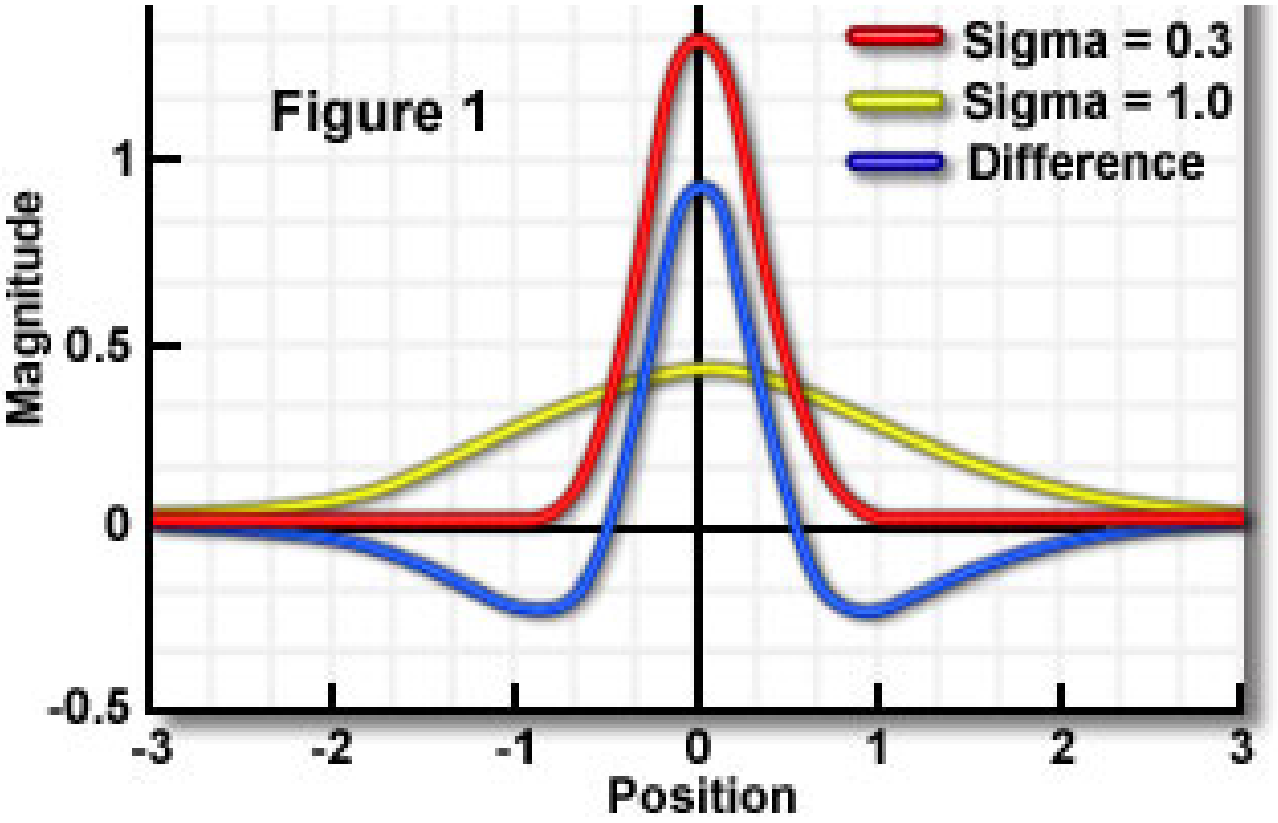
Spatial Filtering

- Image derivatives
  - Laplacian of Gaussian (LoG)
    - (1) “Blob” detector
      - Think: correlation with hat shape
    - (2) Gives strong positive and negative responses at edge
    - **Left**: 3x3 Gaussian + 3x3 Laplacian versus **Right**: 5x5 masks



Spatial Filtering

- Image derivatives
  - **Difference of Gaussian (DoG)**
    - Resembles (negative of) LoG when variances are carefully chosen

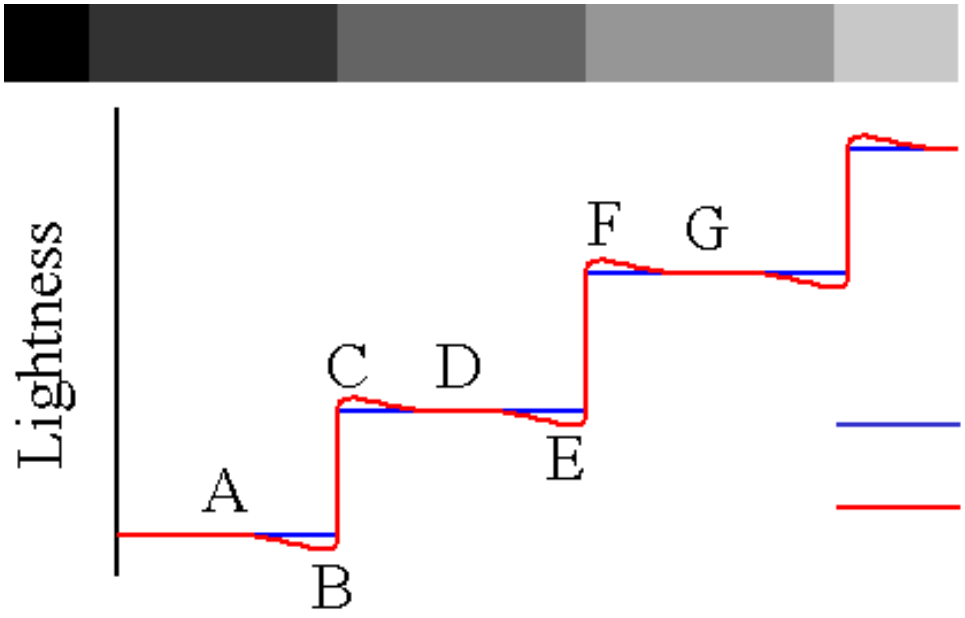


Spatial Filtering

- Sharpening
  - **Unsharp masking**
    - Unsharp ← blur
    - Mask ← negative
  - Common to apply to photos before printing
    - Because printed output is a blurred version of input
    - Sharpening helps compensate printer limitations

Spatial Filtering

- Sharpening
  - Unsharp masking
  - Relates to visual perception → Mach bands

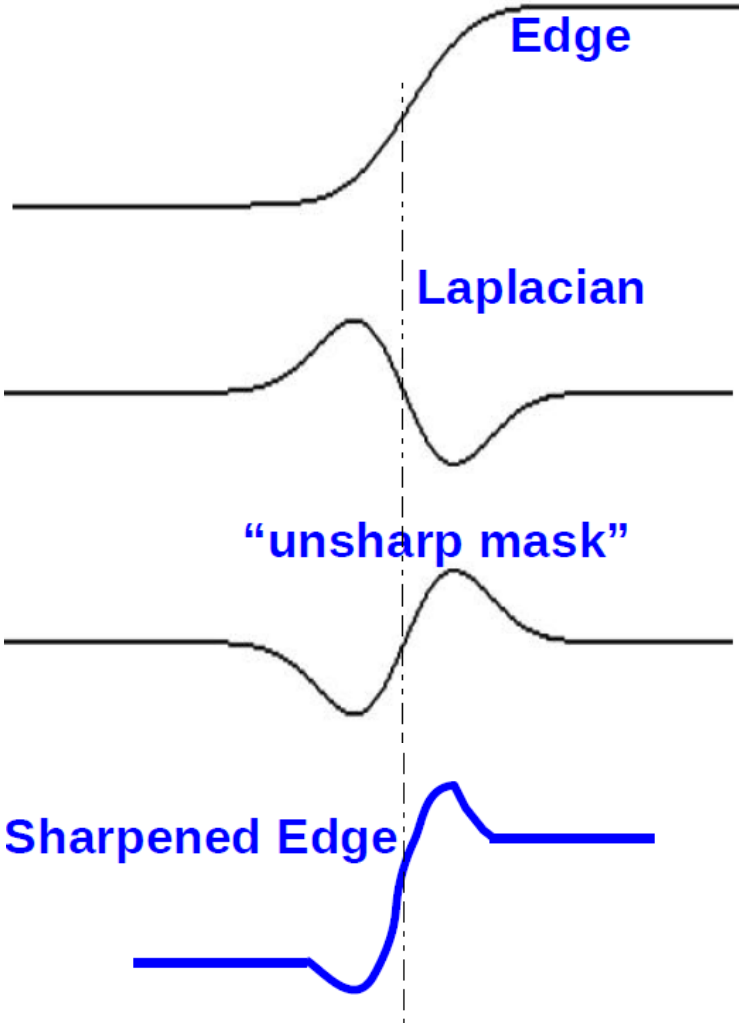


Spatial Filtering

- Sharpening
  - Unsharp masking

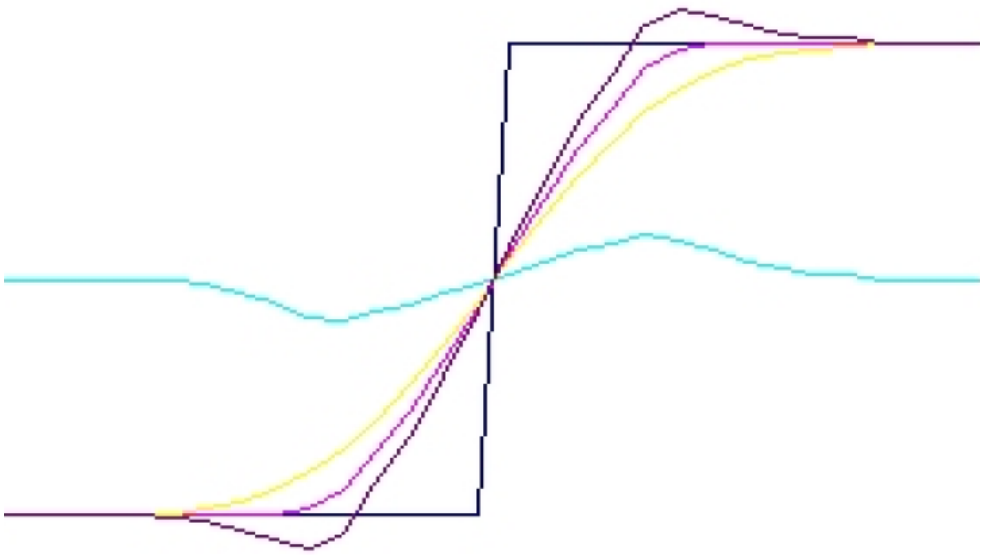
Spatial Filtering

- Sharpening
  - How is unsharp masking related to image derivatives ?
    - Input = Image F (already a bit blurred)
    - Further blur input image → G * F,
      - G = Gaussian mask, * = convolution
      - "Unsharp"
    - Negated-blurred image = – G * F
    - Subtract negated-blurred image from F
      → F – G * F
      - Unsharp "mask"
        (DoG = – LoG of step edge)
    - Unsharp-mask filter
      = F + s (F – G * F)
      ~ F + s DoG * F
      ~ F – s LoG * F
      ~ F – scaled-Laplacian-F



Spatial Filtering

- Sharpening
  - Unsharp masking



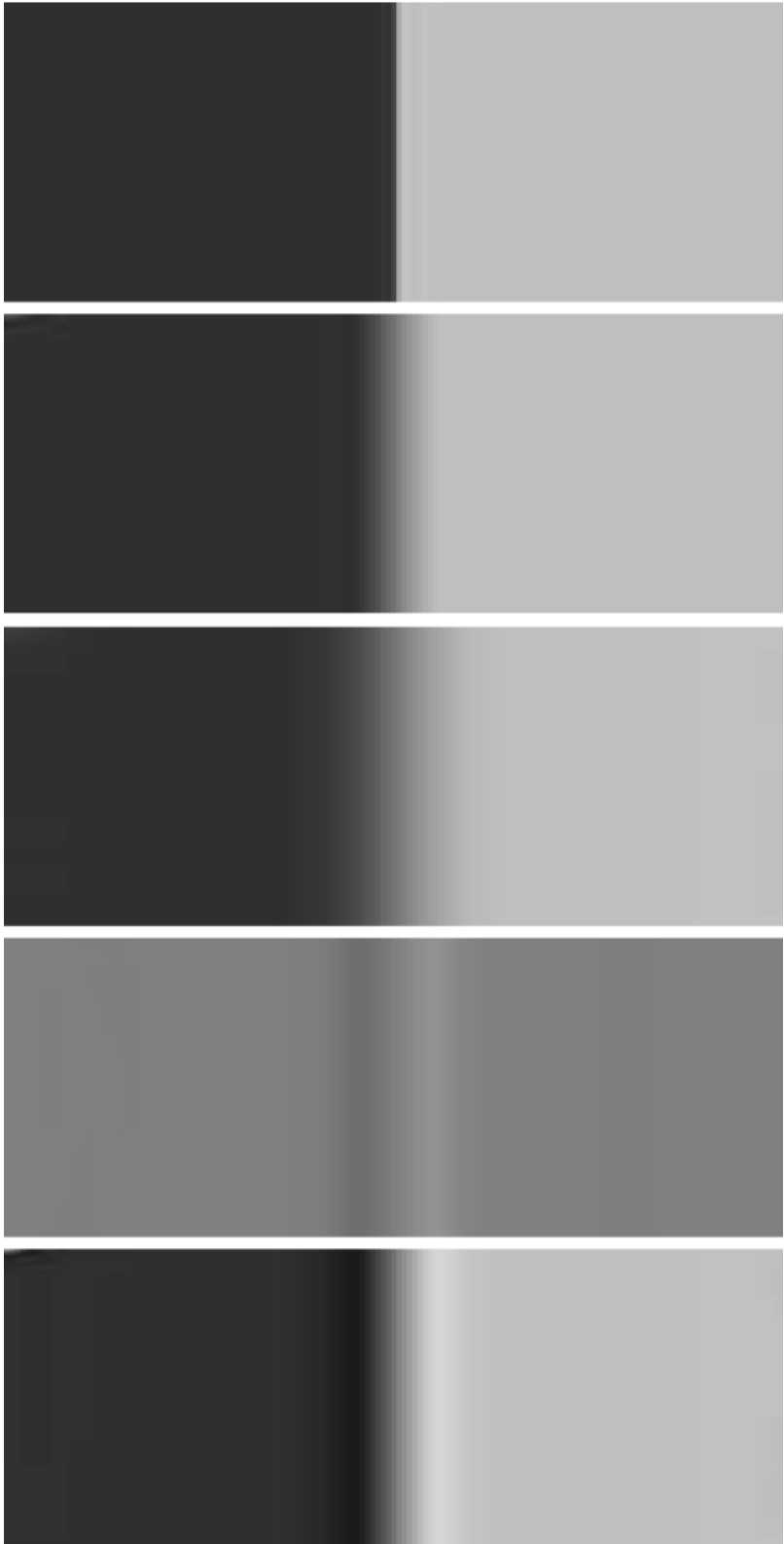
A) Ground Truth

B) Acquired

C) Unsharped

D) Acquired – Unsharped
= DoG (Truth)

E) Result



Spatial Filtering

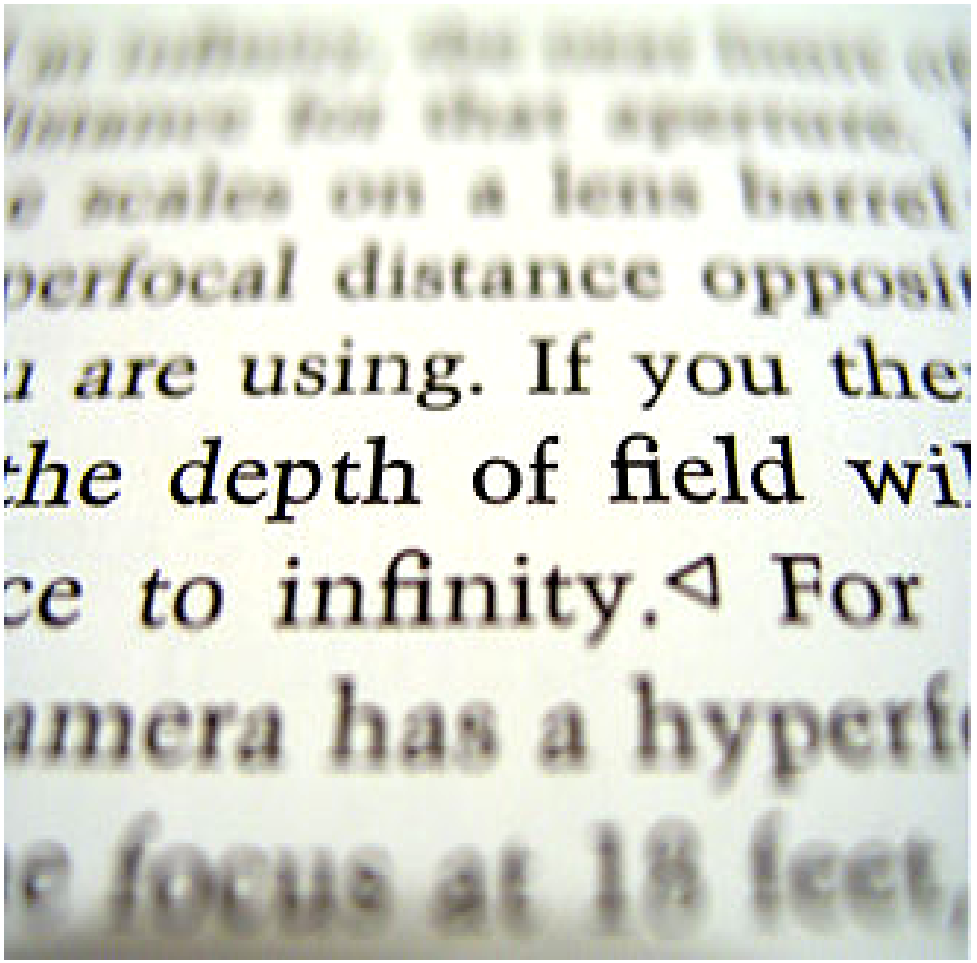
- 1st derivatives versus 2nd derivatives
  - 1st derivatives used for:
    - Edge detection
  - 2nd derivatives used for:
    - Sharpening image (via LoG or DoG)
    - Blob detection

Spatial Filtering

- Applications in photography
  - Bokeh = aesthetic quality of the blur produced in the out-of-focus parts of an image, by a lens
  - In Japanese, “boke” = blur, haze

Spatial Filtering

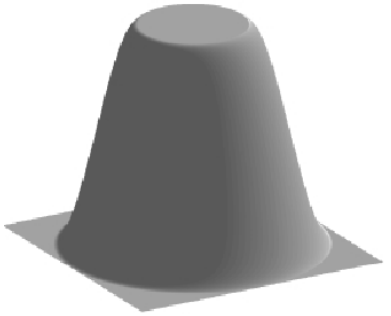
- Applications in photography
  - Bokeh
    - Desire "**depth of field**" for artistic effects
    - Main subject should be in focus. All else should be out of focus.

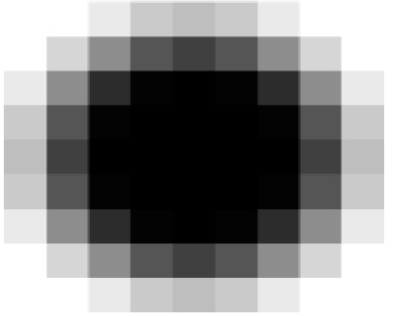


Spatial Filtering

- Applications in photography
  - Bokeh effects in art

Spatial Filtering





- Applications in photography
  - Bokeh simulated in software in part of image
    (1) original (2) bokeh background (3) Blurred bg

Spatial Filtering

- Boundary conditions
  - What happens when mask is placed to pixel close to image boundary ?
  - What are pixel intensities outside image boundary ?

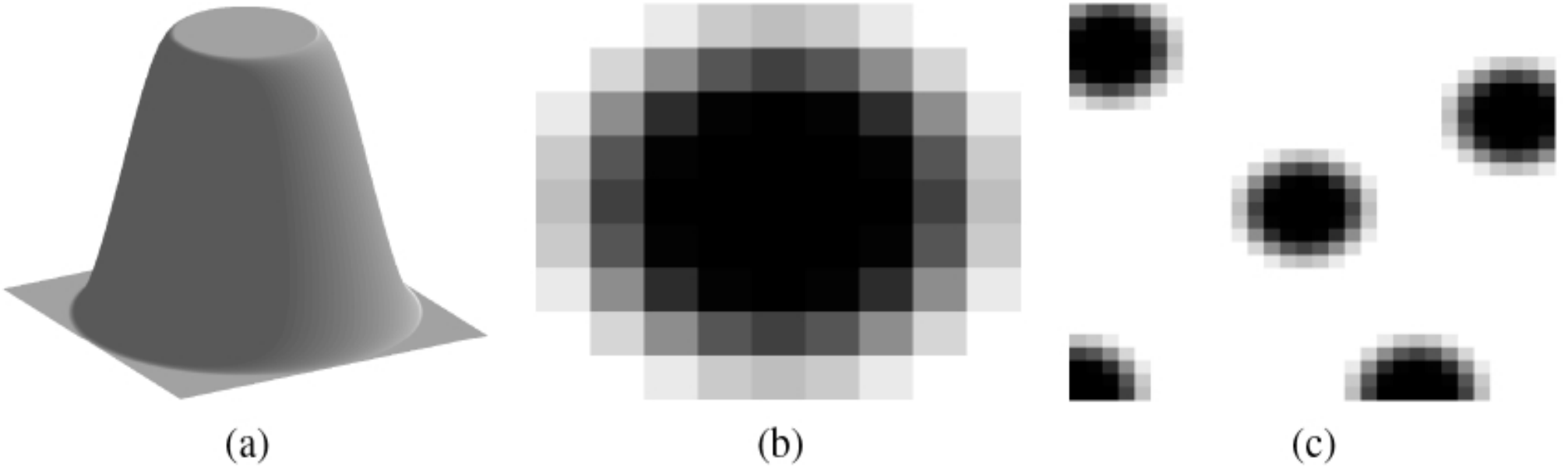


Figure 3.3. Neighborhood shapes. (a) Preserving rotational invariance via a neighborhood mask consisting of a flat central circular plateau with cubic splines on the sides. (b) The discrete sampling of the mask (black $\equiv$ 1, white $\equiv$ 0) for a 9×9 pixels neighborhood. (c) Anisotropic neighborhoods at boundaries.

Spatial Filtering

- Boundary conditions
  - What are pixel intensities outside image boundary ?
    - (1) Zero, or any fixed value
      - In the field of PDEs, Dirichlet / fixed boundary condition
    - (2) Use intensity of the closest pixel inside image
      - Normal derivative of function on boundary = 0
        - Derivative along a direction normal to the boundary
      - In the field of PDEs, Neumann boundary condition
    - (3) Use intensity of pixel determined by circular wrap around
      - Only makes sense in special cases when function is periodic
    - (4) Crop the mask (redefine the mask)
      - Redefine the neighborhood at each pixel