Lecture 4
Image Processing

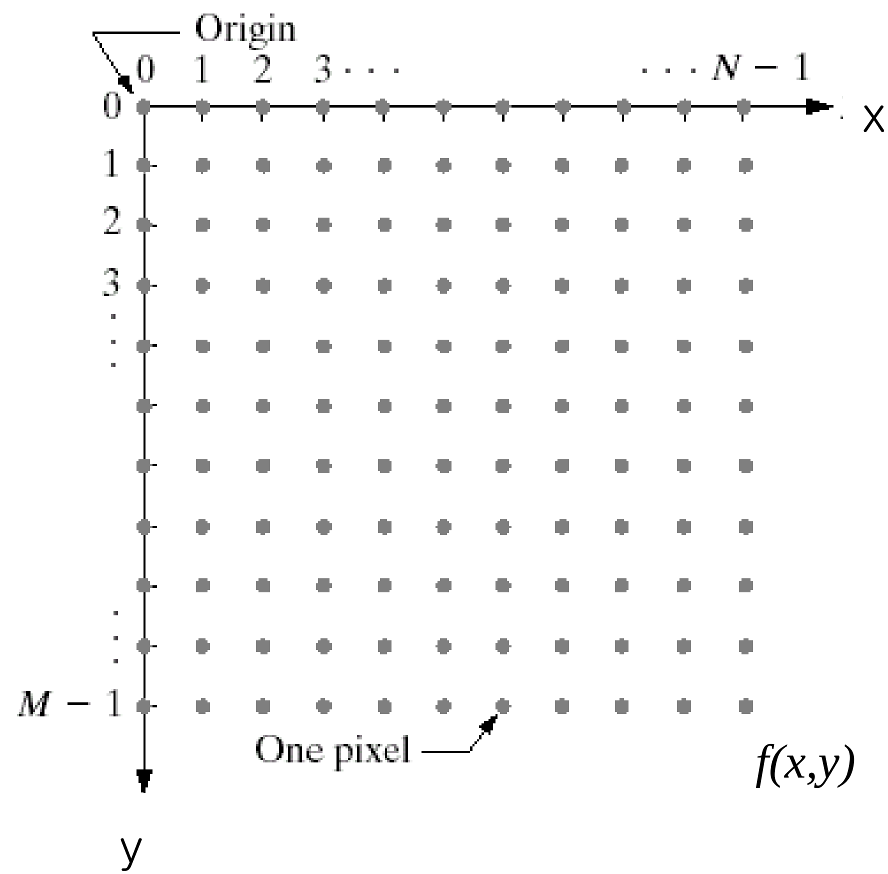
Multimedia Systems
Spring 2020

Digital Images

- 2D matrix of *pixels*.
  - A pixel (Picture Element) is the smallest unit of color in a digital image.
  - Number of pixels determines the *resolution*.
- Grey Scale Images:
  - A single value corresponding to a grey level.
  - 8 bits is usually used to represent a grey scale
- Color Images
  - Three values corresponding to some RGB levels
    - Red Green and Blue (RGB)
    - 24 bits to represent a color

Coordinates in an image

- Image representation
  -> 2D function
- f(x,y)
  - f(0,0) : 1st col. 1st row
  - f(1,0) : 2nd col. 1st row
  - f(0,1) : 1st col. 2nd row



Coordinates in an image

f(100,70)

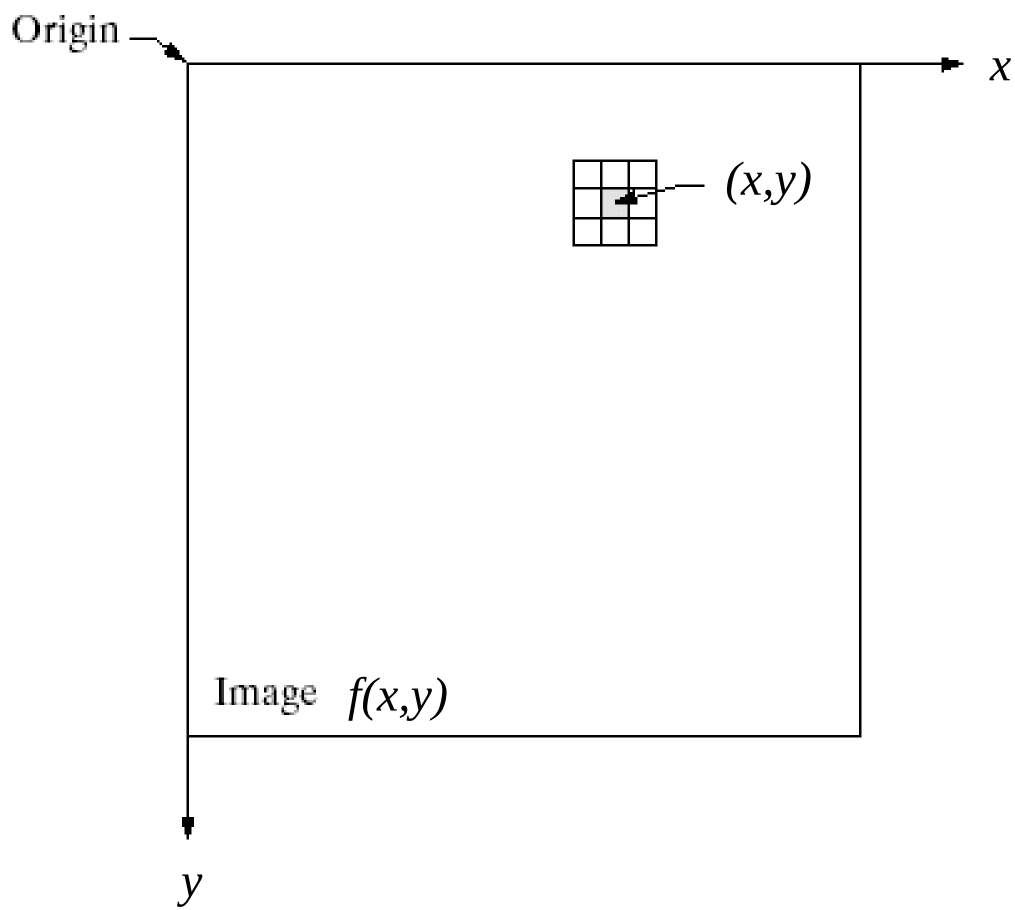
Y \ X	100	101	102	103	104
70	5	10	158	160	160
71	4	23	140	150	157
72	3	17	138	145	158
73	3	15	120	138	145
74	2	15	122	139	143

f(104,74)

Point and Neighborhood

- 3x3 neighborhood about a point (x,y) in an image

  point f(x,y)
- 8 neighborhood f(x−1,y−1), f(x,y−1), f(x+1,y−1), f(x−1,y), f(x+1,y), f(x−1,y+1), f(x,y+1), f(x+1,y+1)



Point Operation (Binarization)

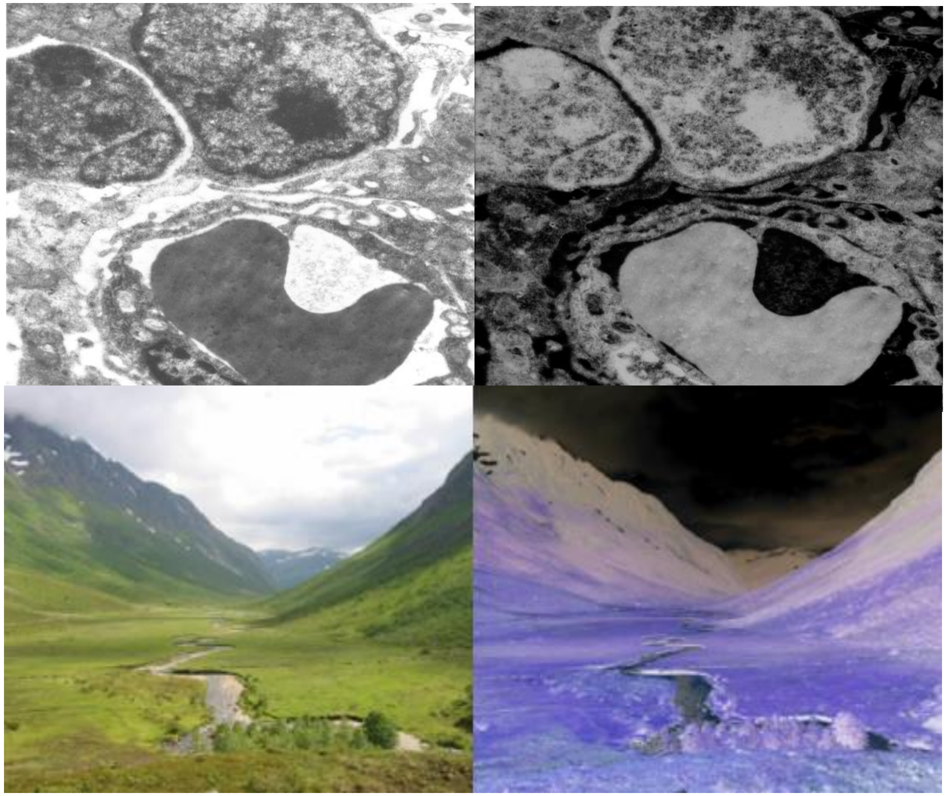
- Point operation: p(x,y) = function(q(x,y))
- Threshold
  - $$p(x,y) = \begin{cases} 255 & \text{if } q(x,y) > \text{threshold} \\ 0 & \text{else} \end{cases}$$

Original

threshold

Point Operation

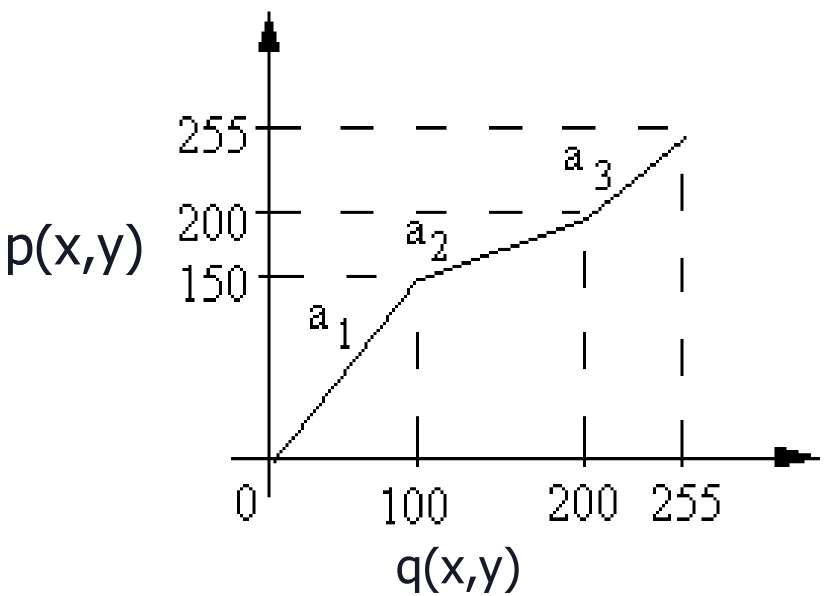
- Image inversion
  - $p(x,y) = 255 - q(x,y)$

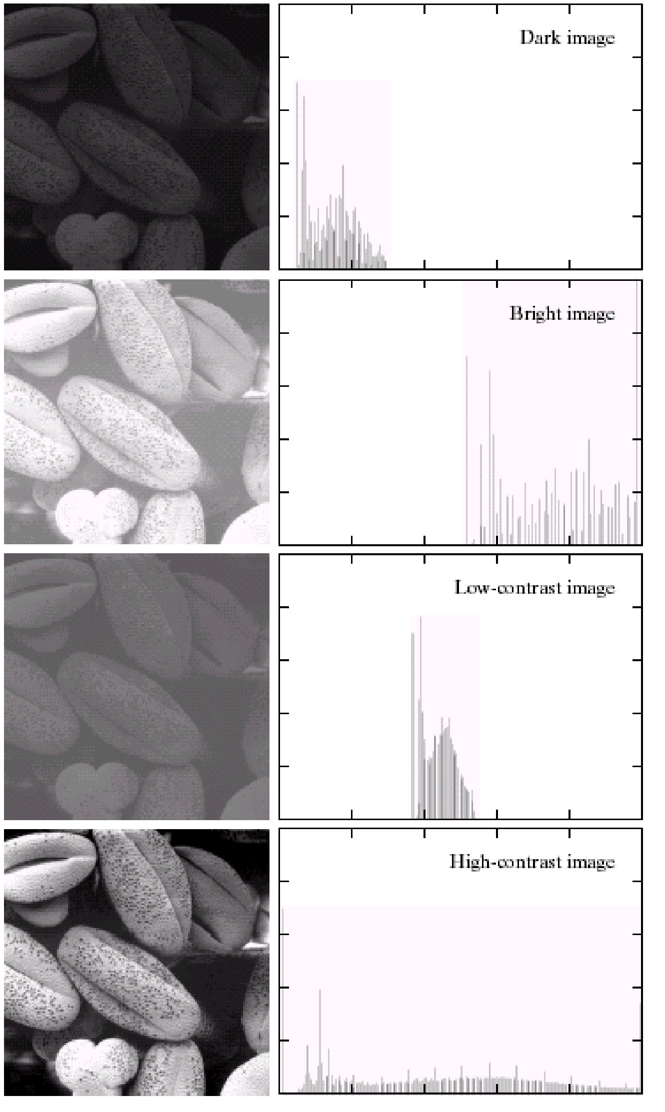


Original Inverted

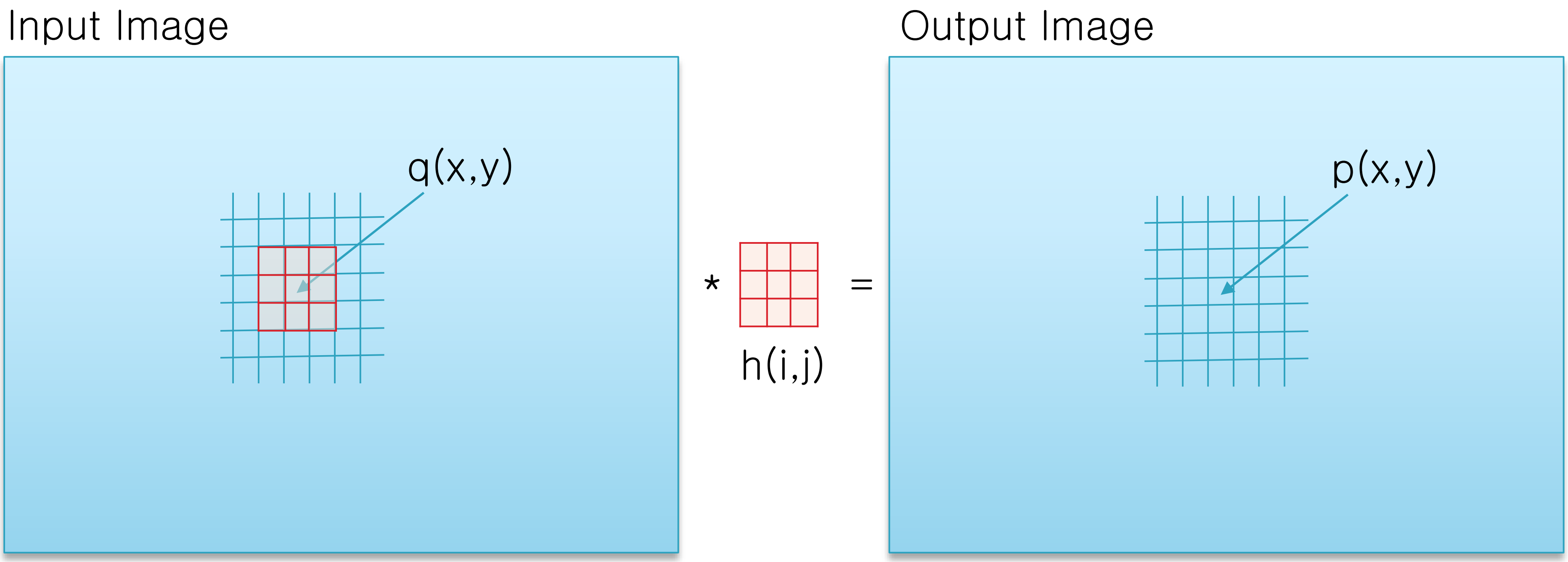
Point Operation

- Histogram Stretching
  - $p(x,y) = a_1\ q(x,y)$ $0 < q(x,y) <= 100$
  - $p(x,y) = a_2\ q(x,y)+100$ $100 < q(x,y) <= 200$
  - $p(x,y) = a_3\ q(x,y)$ $200 < q(x,y) <= 255$





Area operation: Image Convolution



Area operation: Image Convolution

- Idea: new pixel values of the image are determined by considering the surrounding pixel values.

$$p(x,y) = q*h = \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} h(j,i)q(x+j,y+i) \quad \text{correlation}$$

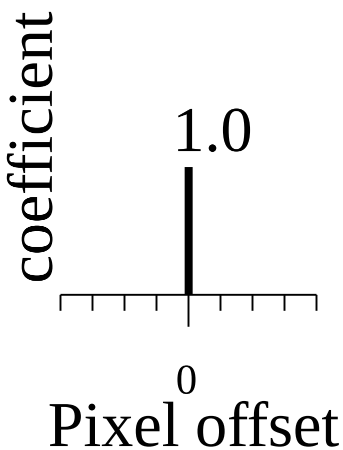
$$p(x,y) = q*h = \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} h(j,i)q(x-j,y-i) \quad \text{convolution}$$

where h is the filter, * denotes convolution, m is the filter size, and m/2 is integer division

Linear filtering

original

q



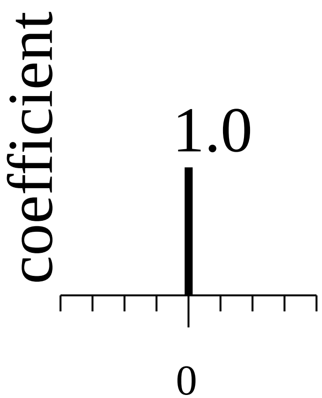
h

?

Slide credit: Bill Freeman

Linear filtering

original



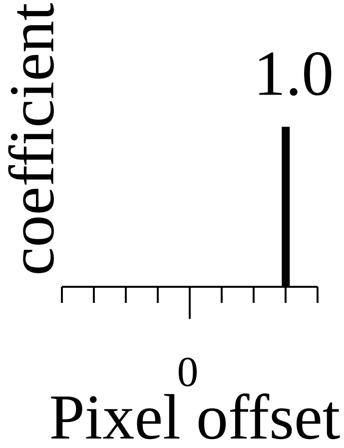
Pixel offset

Filtered
(no change)

Slide credit: Bill Freeman

Linear filtering

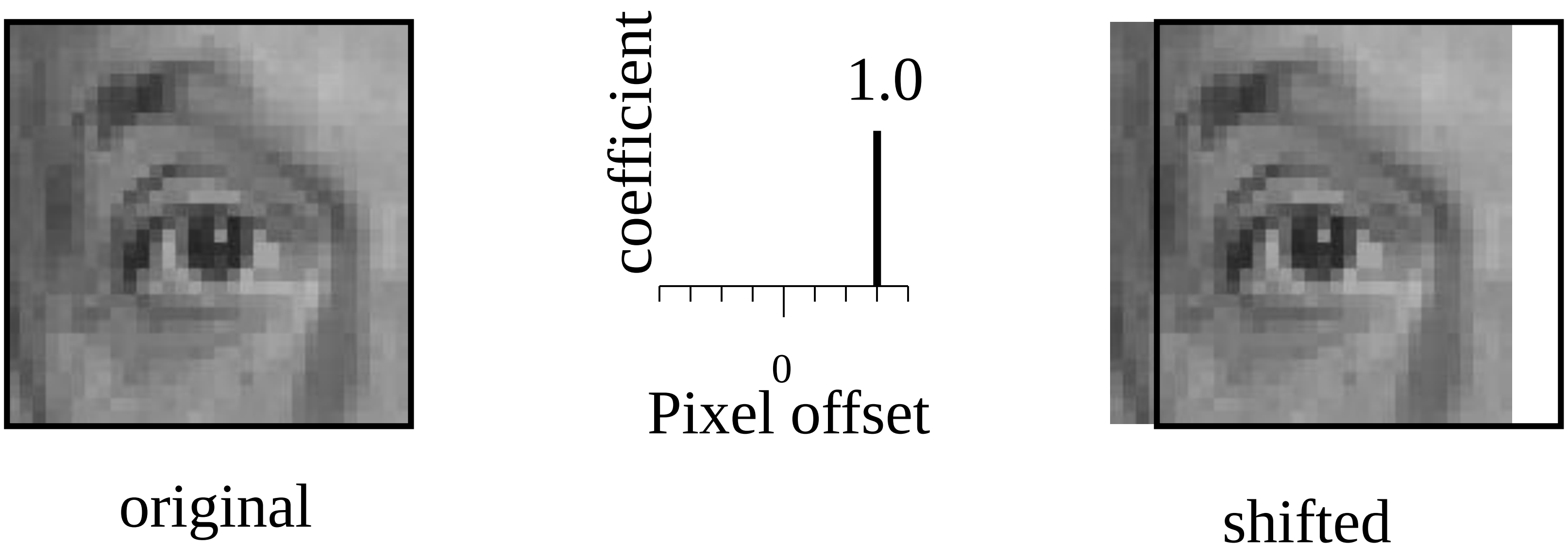
original



?

Slide credit: Bill Freeman

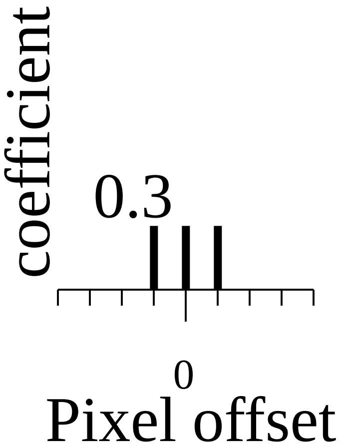
shift



Slide credit: Bill Freeman

Linear filtering

original

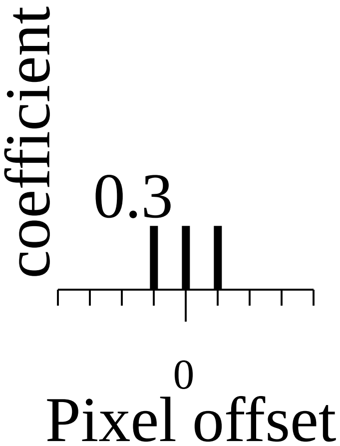


?

Slide credit: Bill Freeman

Blurring

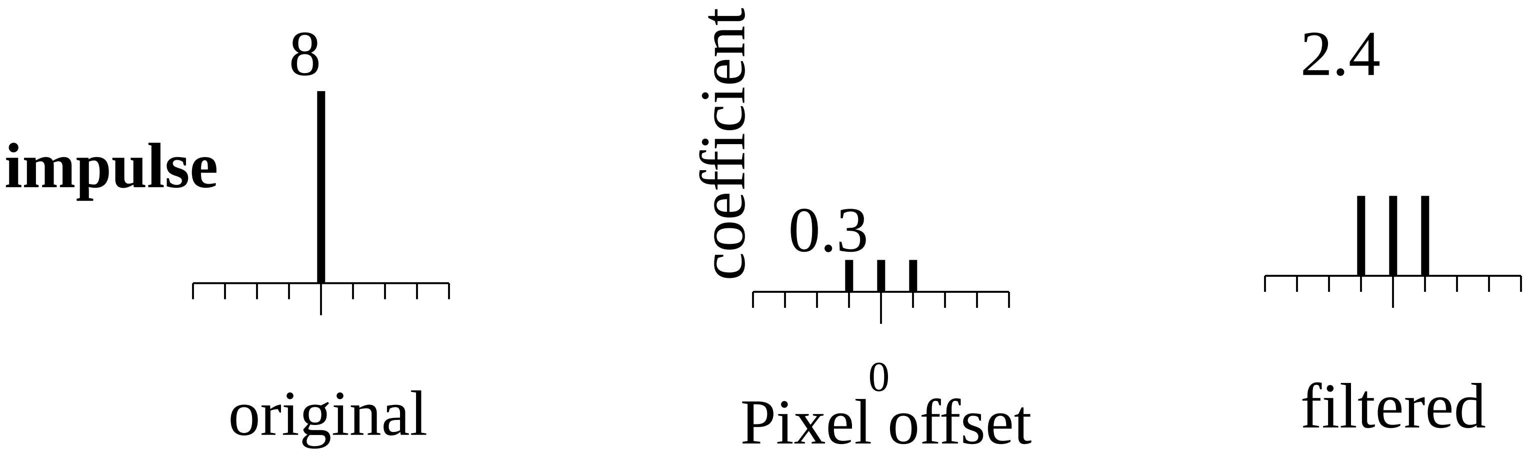
original



Blurred (filter applied in both dimensions).

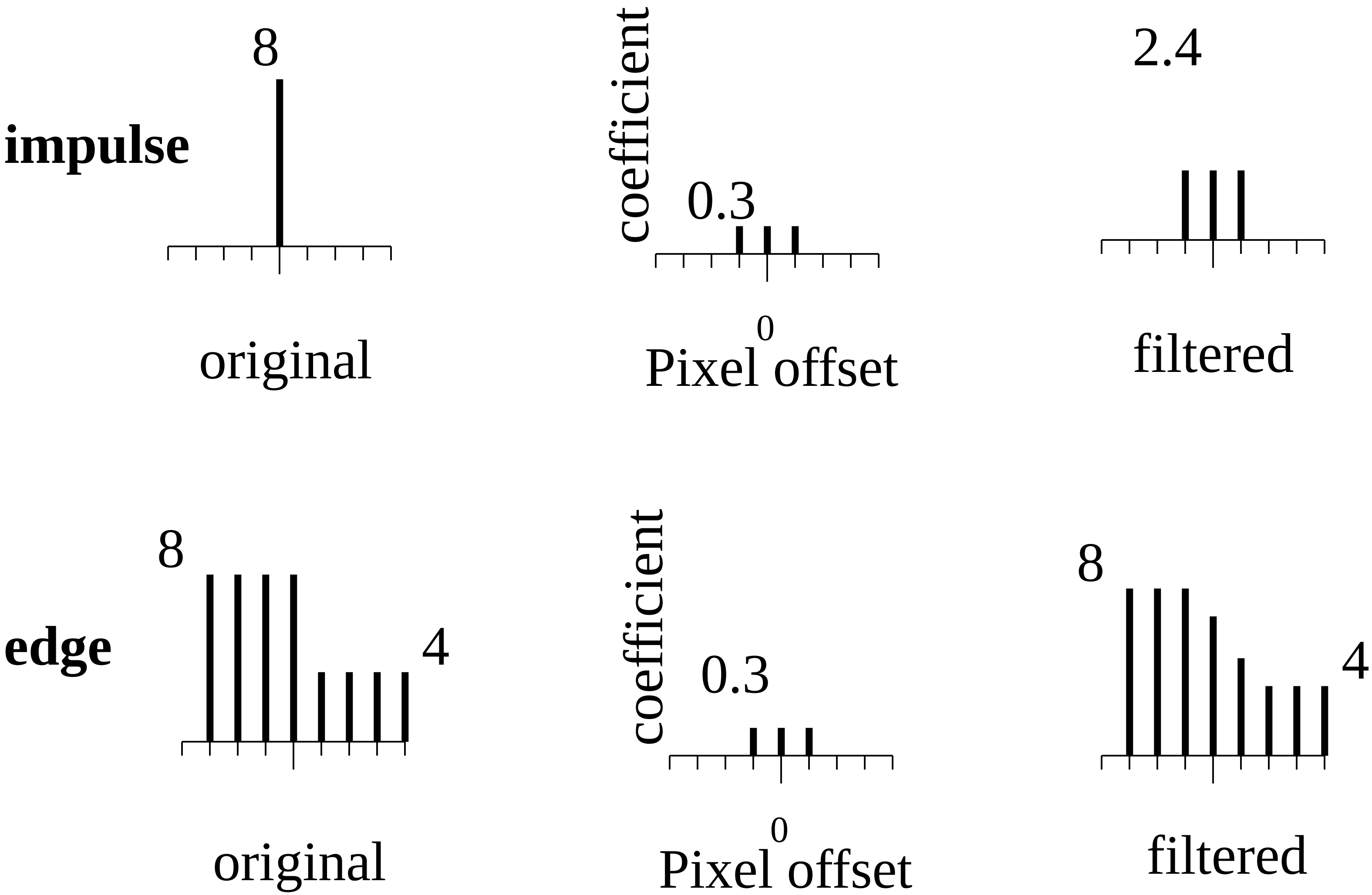
Slide credit: Bill Freeman

Blur examples



Slide credit: Bill Freeman

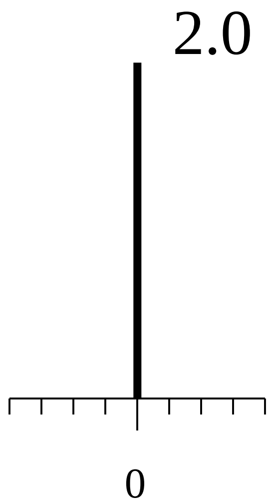
Blur examples



Slide credit: Bill Freeman

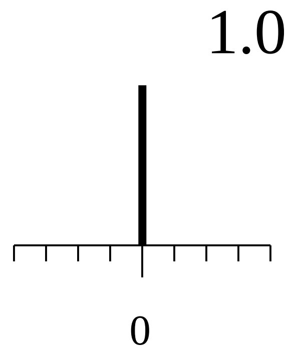
Linear filtering

original



h_1

$-$

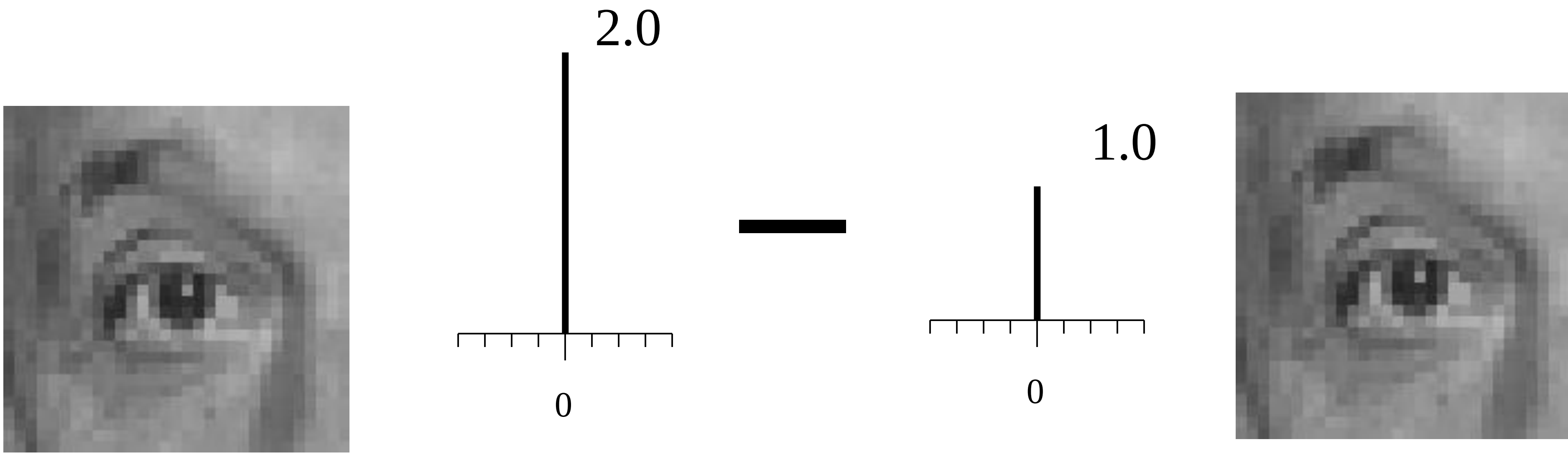


h_2

?

Slide credit: Bill Freeman

Linear filtering (no change)



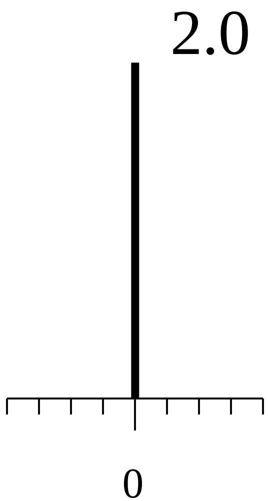
original

Filtered
(no change)

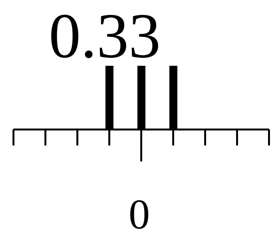
Slide credit: Bill Freeman

Linear filtering

original



−



?

Slide credit: Bill Freeman

(remember blurring)

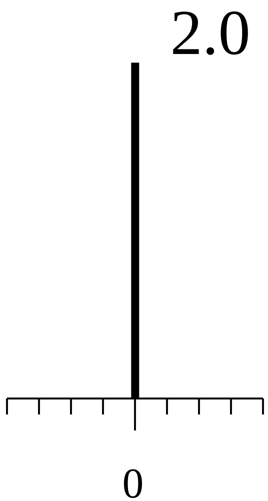
original

Blurred (filter applied in both dimensions).

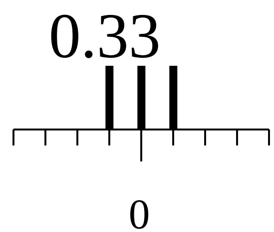
Slide credit: Bill Freeman

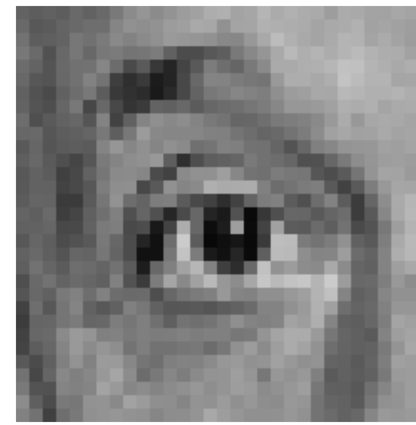
Sharpening

original



−

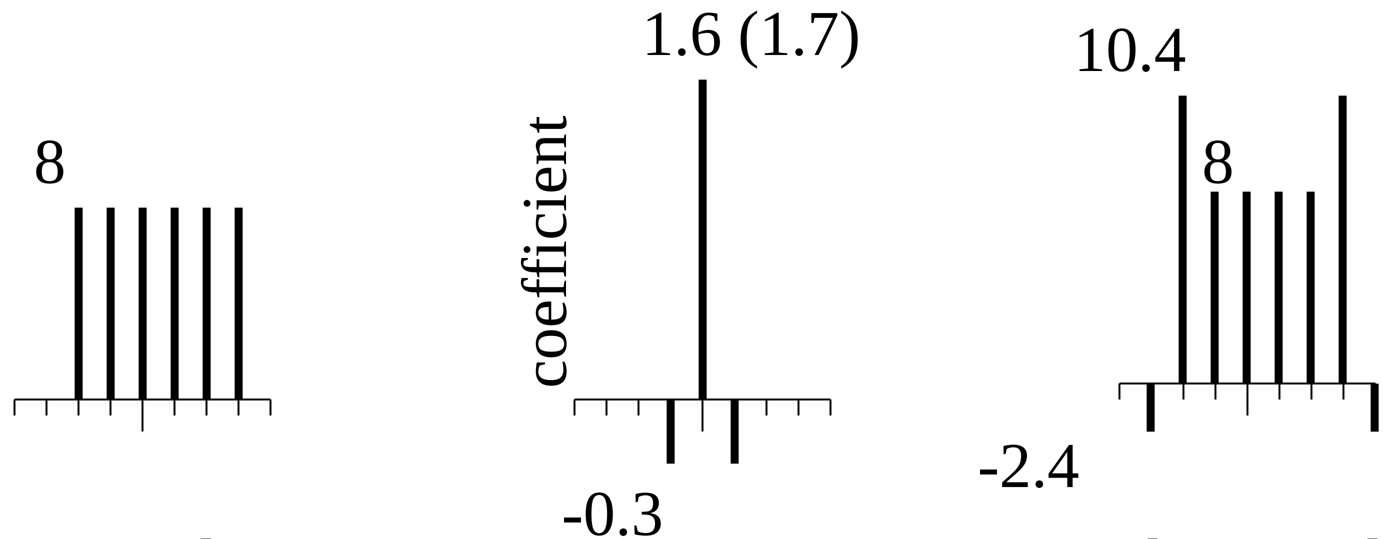




Sharpened
original

Slide credit: Bill Freeman

Sharpening example

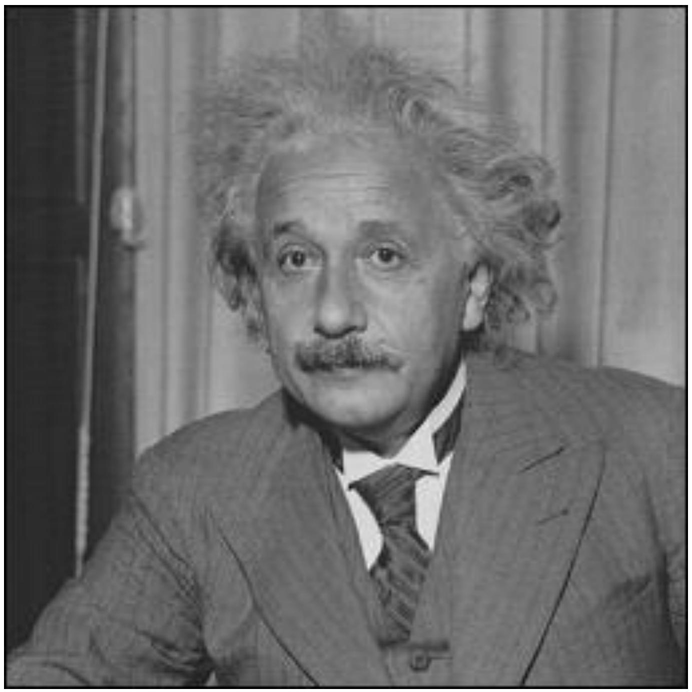


original

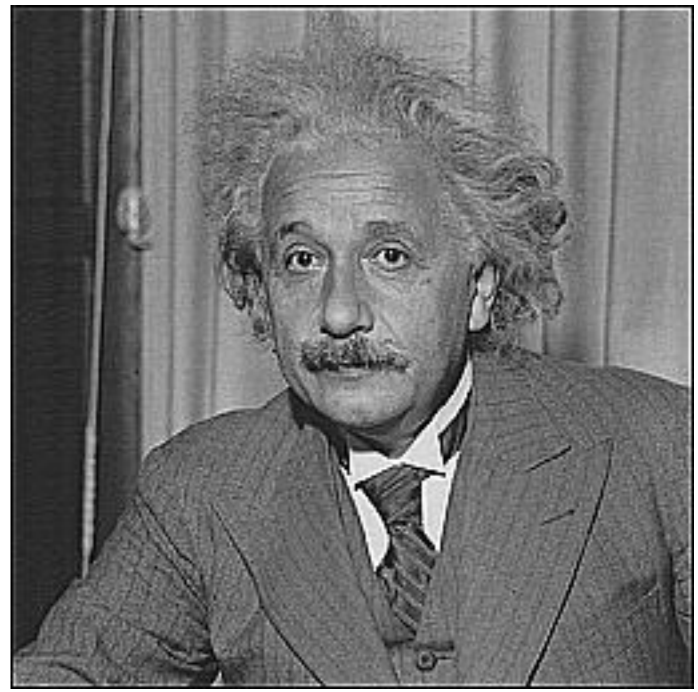
Sharpened
(differences are
accentuated; constant
areas are left untouched).

Slide credit: Bill Freeman

Sharpening



before



after

Slide credit: Bill Freeman

Spatial filtering

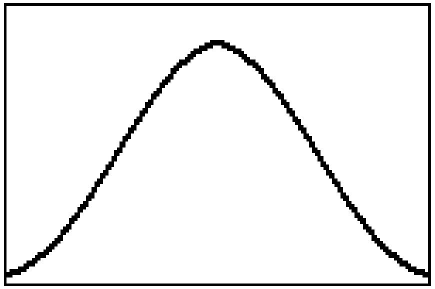
- Representation of a general 3x3 spatial filter mask

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Image Convolution

- **Average Smoothing:**
  - $$A_{avg} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Gaussian Smoothing:**
  - $$A_{Gaus} = \frac{1}{106}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 9 & 9 & 9 & 1 \\ 1 & 9 & 18 & 9 & 1 \\ 1 & 9 & 9 & 9 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

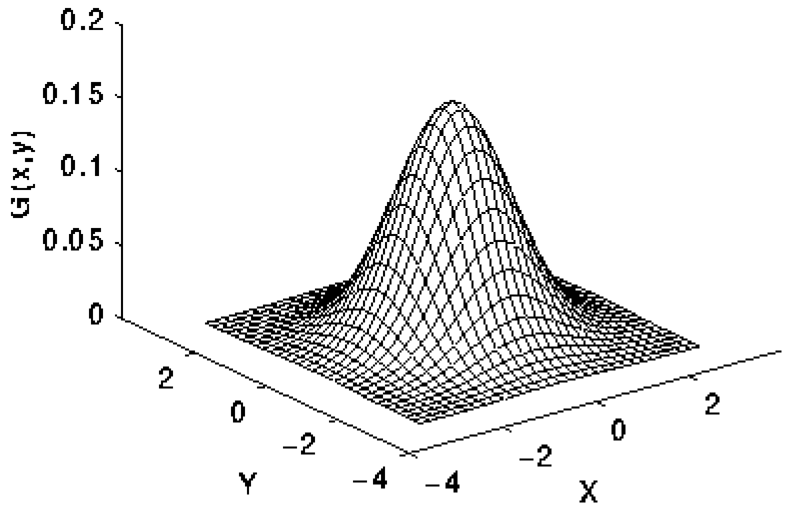


Gaussian filter

- Smoothing (lowpass) filter
- The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise
- 1D Gaussian example (sigma=1.0)

0.005	0.054	0.242	0.398	0.242	0.054	0.005

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



2D Gaussian filter (sigm1=1.0)

Gaussian filter

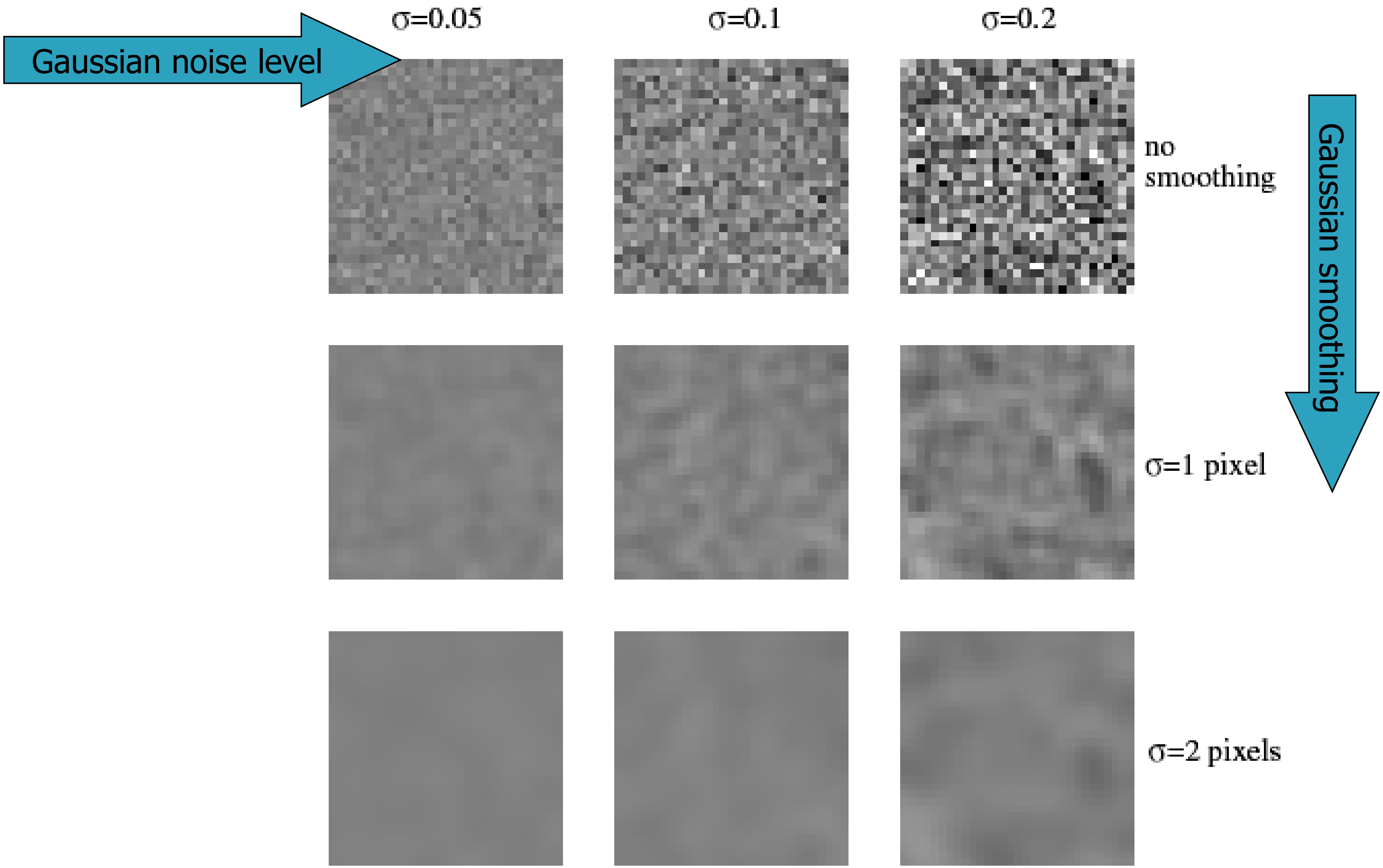


Image Convolution Example

- **Average Smoothing with 3x3 filter:**

q(x,y)

2	1	2	1
0	25	1	1
1	2	2	1
2	1	2	1

* 1/9

h(x,y)

1	1	1
1	1	1
1	1	1

=

p(x,y)

2	1	2	1
0	4	4	1
1	4	4	1
2	1	2	1

$$p(x,y) = q * h = \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} h(i,j)\, q(x-i, y-j)$$

Average Smoothing

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	$\boxed{10}$	10	0	0	0	0	0	

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Image Convolution Example

Gaussian blur filter effect

Sharpening

mask $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$ *or* $\begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$ *or* $\begin{pmatrix} -k & -k & -k \\ -k & 8k+1 & -k \\ -k & -k & -k \end{pmatrix}$

Edge Detection

- Derivative of a linear function f(x,y)

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon} \right)$$

- We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

- In image, this can be
  - I(x+1,y) – I(x,y) or I (x+1,y) – I(x−1,y)

Simple Edge Detection

- **Simple Edge Detection Idea:**

1. $\partial I/\partial x$

   Simply convolve Image with [1 0 –1]

   Resulting image: I_x

2. $\partial I/\partial y$

   Simply convolve Image with $[1\ 0\ -1]^T$

   Resulting image: I_y

3. ***Image Gradient (magnitude** $\partial I/\partial x$ and $\partial I/\partial y$**)** :

   $\nabla I(i,j) = \sqrt{[I_x(i,j)^2 + I_y(i,j)^2]}$

Input Image

0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	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
0	0	0	0	0	0	0	0	0	0

$$* \quad \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \quad =$$

$\partial I/\partial x$

0	0	0	0	-9	-9	0	0	0	0
0	0	0	0	-9	-9	0	0	0	0
0	0	0	0	-9	-9	0	0	0	0
0	0	0	0	-9	-9	0	0	0	0
0	0	0	0	-9	-9	0	0	0	0
0	0	0	0	-9	-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	0	0	0	0
0	0	0	0	0	0	0	0	0	0

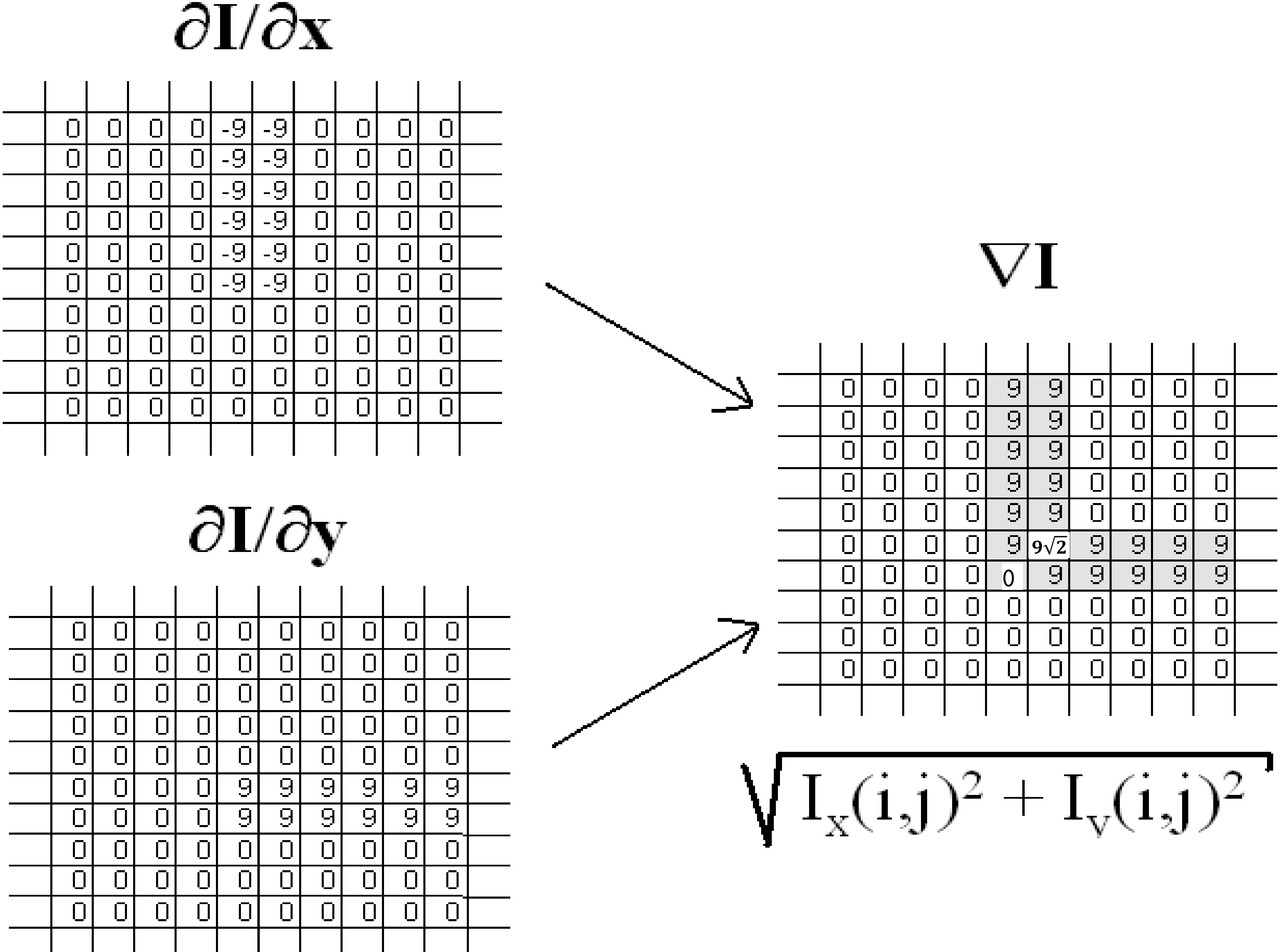
Input Image

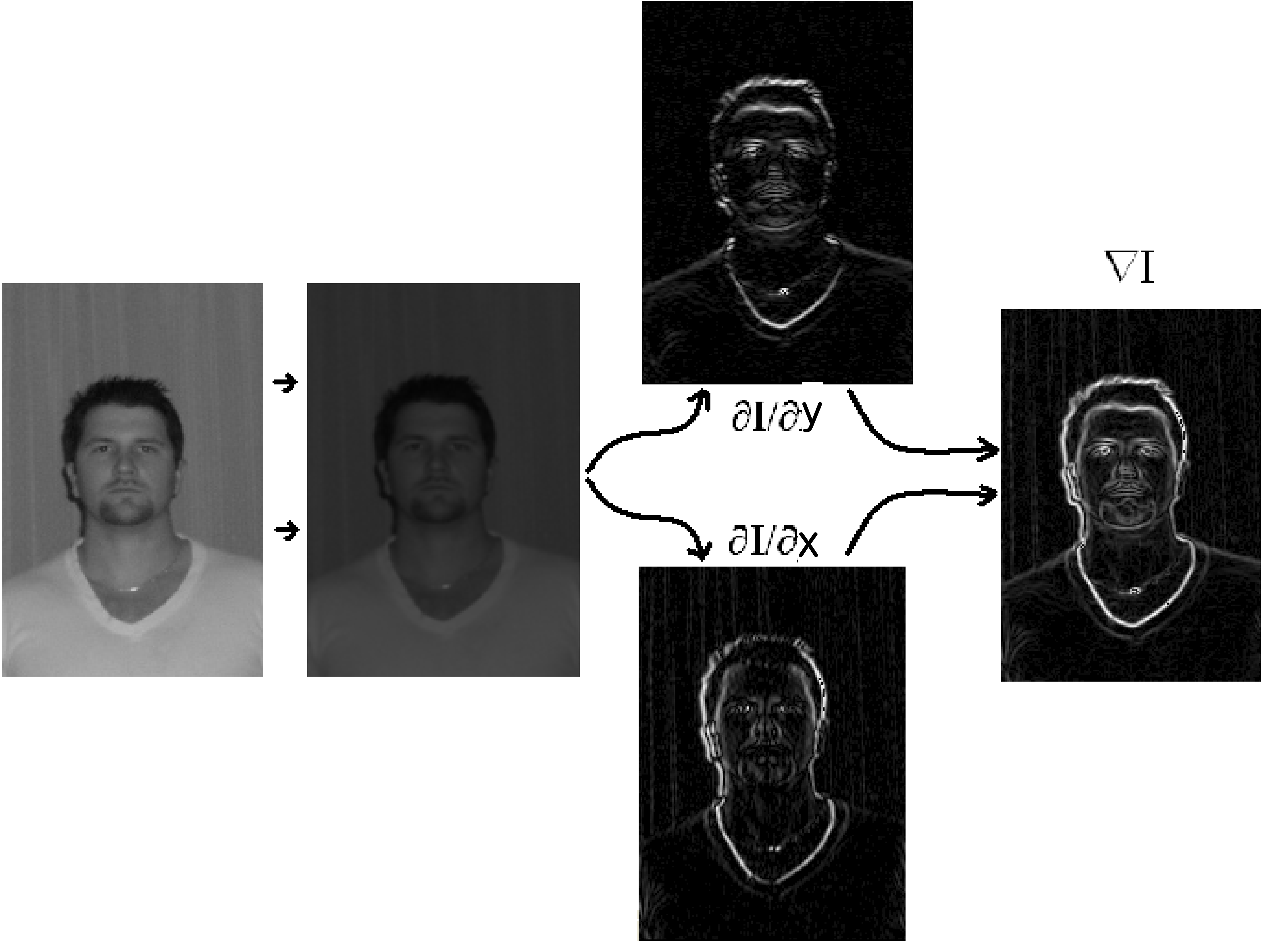
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$* \quad \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad =$$

$\partial I/\partial y$

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
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	9	9	9	9	9
0	0	0	0	0	9	9	9	9	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


∂I/∂x
∂I/∂y
∇I
$\sqrt{I_x(i,j)^2 + I_y(i,j)^2}$



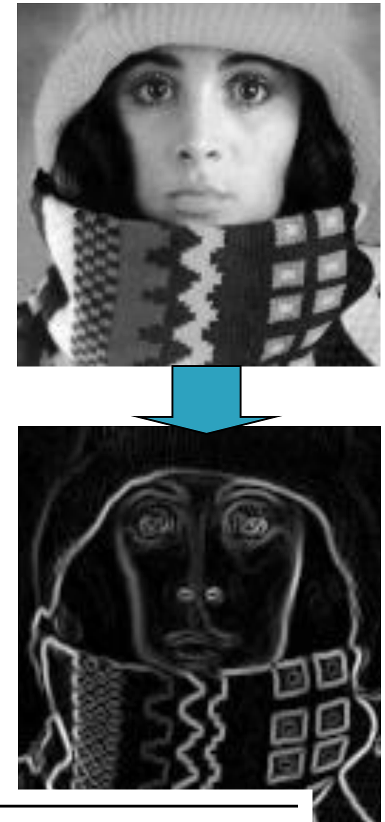
$\partial I/\partial y$
$\partial I/\partial x$
∇I

Sobel Edge Detection

- Sobel edge filter

$$[\mathbf{h}_x] = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$[\mathbf{h}_y] = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

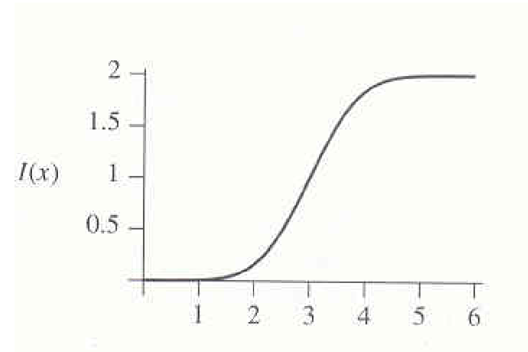


$$p(x,y) = \sqrt{(q(x,y) * h_x(x,y))^2 + (q(x,y) * h_y(x,y))^2}$$

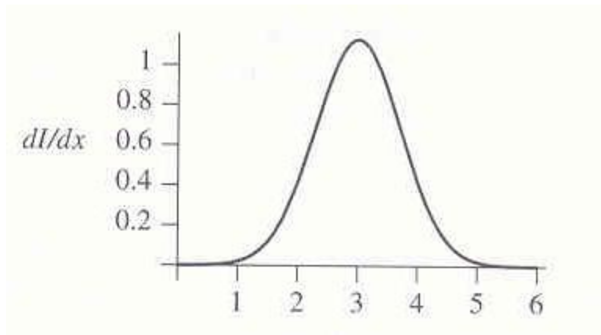
$$\theta = \arctan\left(\frac{q(x,y) * h_x(x,y)}{q(x,y) * h_y(x,y)}\right)$$

Laplacian Filters

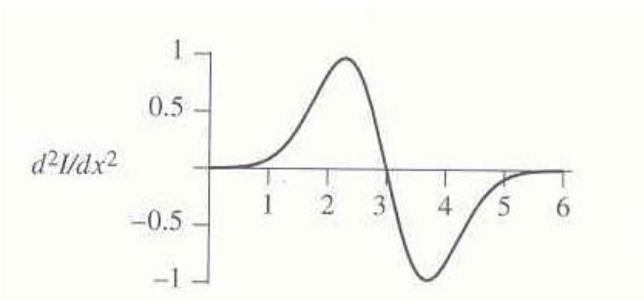
- Let us take a look at the one-dimensional case:
- A change in brightness:



- Its first derivative:



- Its second derivative:



Laplacian Filters

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

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial G_x}{\partial x} = \frac{\partial(f[i, j+1] - f[i, j])}{\partial x}$$

$$= \frac{\partial(f[i, j+1]}{\partial x} - \frac{\partial f[i, j]}{\partial x}$$

$$= (f[i, j+2] - f[i, j+1]) - (f[i, j+1] - f[i, j])$$

$$= f[i, j+2] - 2 f[i, j+1] + f[i, j]$$

Laplacian Filters

- Since we want the computation centered at [i,j]:

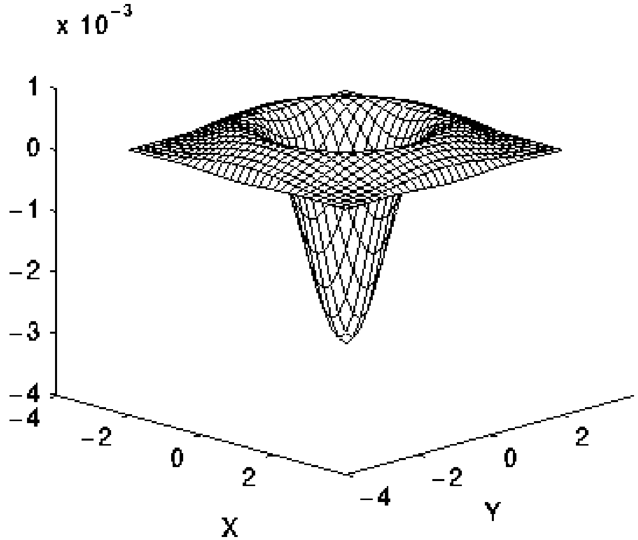
$$\frac{\partial^2 f}{\partial x^2} = f[i, j+1] - 2f[i, j] + f[i, j-1]$$

$$\frac{\partial^2 f}{\partial y^2} = f[i+1, j] - 2f[i, j] + f[i-1, j]$$

- Obviously, each function can be computed using a 1, -2, 1 convolution filter in x or y direction, respectively.
- Since ∇^2 is defined as the sum of these two functions, we can use a single 3×3 (or larger) convolution filter for its computation (next slide).

Laplacian Filters

- Continuous variant:



Discrete variants
(applied after smoothing):

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

Laplacian of Gaussian

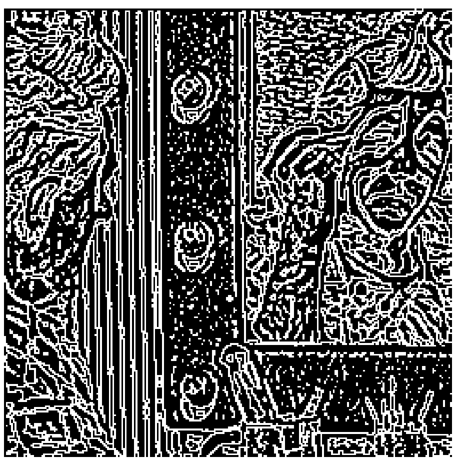
- Bad idea to apply a Laplacian without smoothing
  - smooth with Gaussian, apply Laplacian
  - this is the same as filtering with a Laplacian of Gaussian (LOG) filter
  - Now mark the zero points *where there is a sufficiently large derivative, and enough contrast*

Laplacian of Gaussian

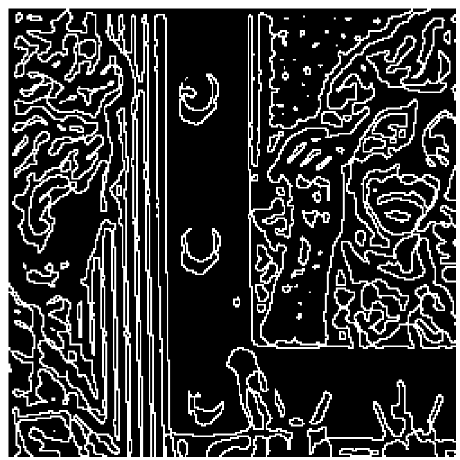
3×3 Laplacian

5×5 Laplacian

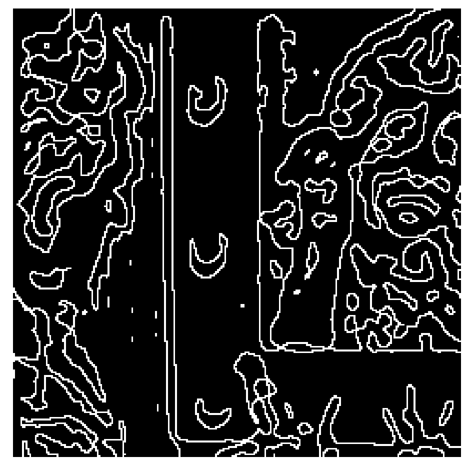
7×7 Laplacian



zero detection



zero detection



zero detection

Median filter

- A **Median Filter** operates over a window by selecting the median intensity in the window.

I(x,y)

4	6	6
6	14	8
7	5	7

median(I(x,y)) = 6

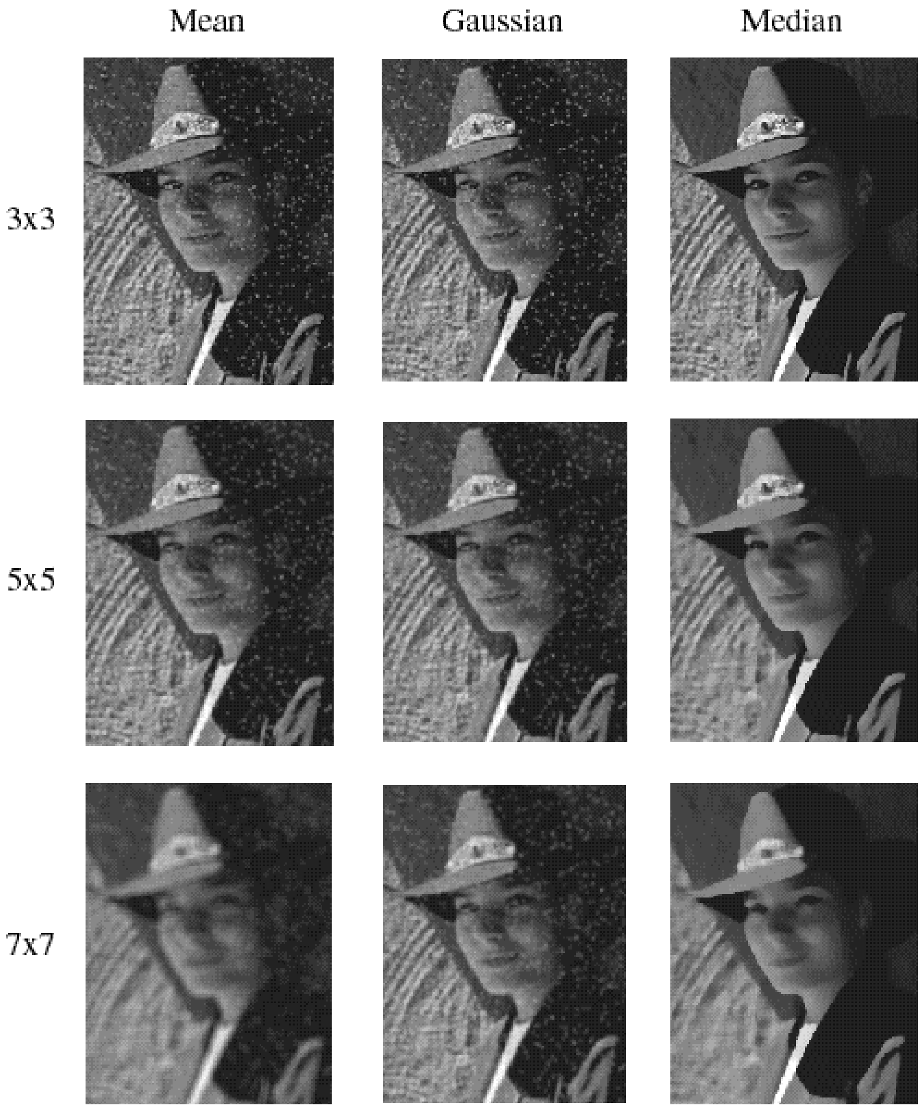
I′(x,y)

4	6	6
6	6	8
7	5	7

- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?
- Median filter is non linear

Median filter

- Useful in removing salt-and-pepper noise
- But, salt-and-pepper noise is not common.
- Also useful in removing noise in a binary image



Template(Block) matching

Template Matching

- Filters can be regarded as templates
  - Applying a filter at some point can be seen as taking a dot-product between the image and some vector
  - Filtering the image is a set of dot products
  - It is a measure of the angle of two vectors (template (block) and image)

Template Matching by Cross-Correlation

- The use of cross-correlation for template matching is motivated by the distance measure (squared Euclidean distance) between a template t(x,y) and image f(x,y)

$$d_{f,t}(x,y) = \sum_{i,j}[t(i,j) - f(x+j,y+i)]^2$$

- The correlation response at each pixel (x,y) is

$$c(x,y) = f * t = \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} t(j,i) f(x+j,y+i)$$

Normalize correlation

- By normalizing the image and feature vectors to unit length, yielding a cosine-like correlation coefficient

$$\gamma(x,y)=\frac{\sum_{i,j}[(t(j,i)-\bar{t})(f(x+j,y+i)-\bar{f}_{x,y})]}{\{\sum_{i,j}(t(j,i)-\bar{t})^2\sum_{i,j}(f(x+j,y+i)-\bar{f}_{x,y})^2\}^{0.5}}$$

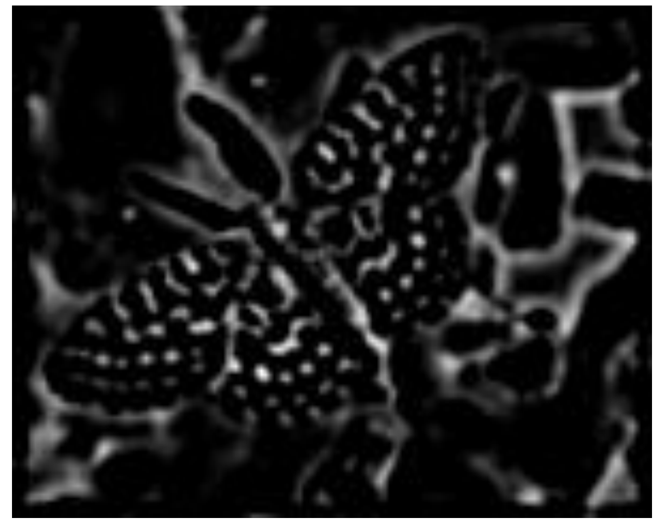
$-1 < \gamma(x,y) <= 1$

- 1 or -1 : exact matching of image and feature
- 0: no correlation

Template matching: example(1)

Template

Original image



Absolute value of correlation
(scale to 0 ~ 255)

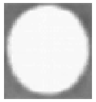
Template matching: example(2)

Template

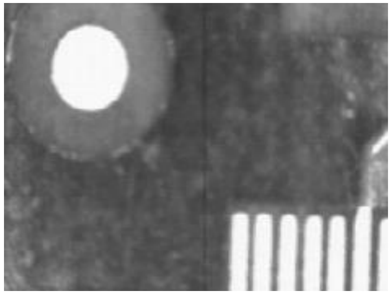
Original image

Absolute value of correlation
(scale to 0 ~ 255)

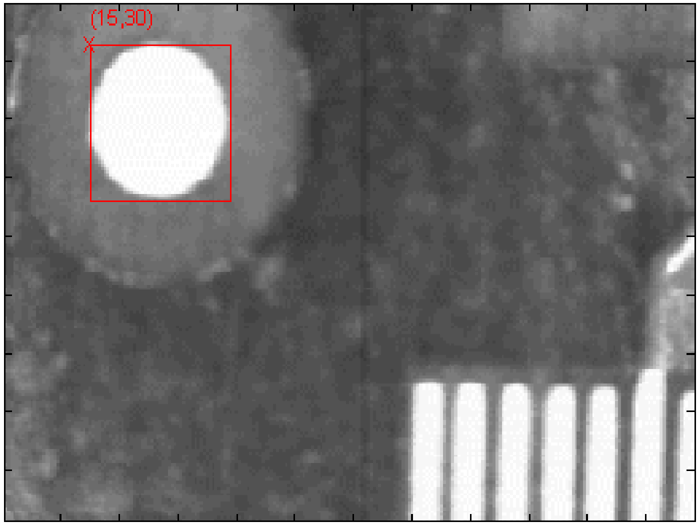
Template matching: example(3)

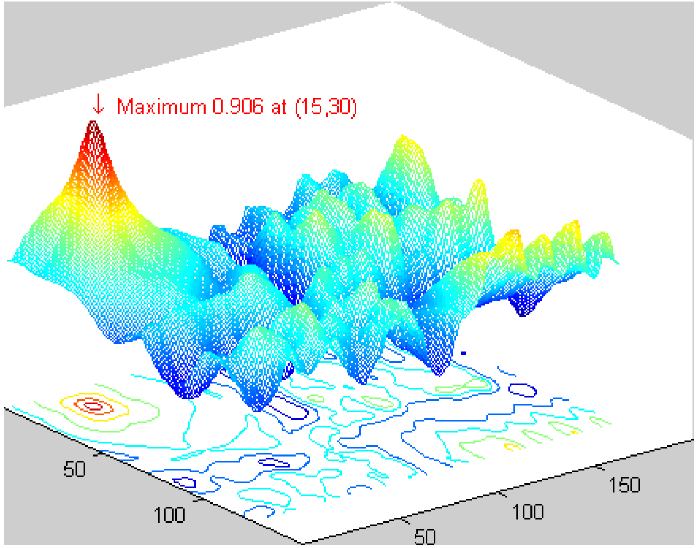


Template



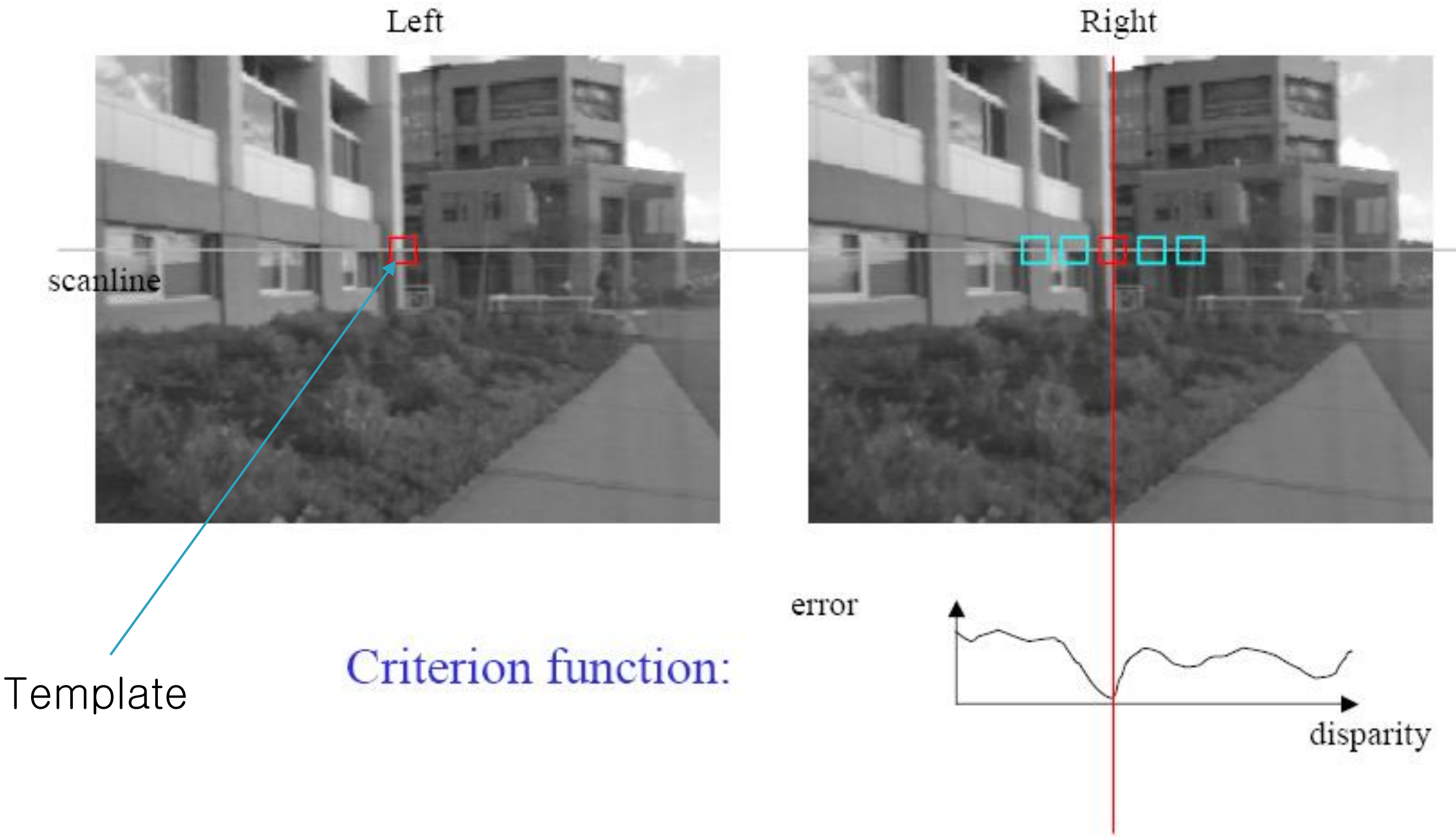
Data Set 3





Normalized correlation

Correspondence by block matching



Criterion function:

Image blocks as a vector

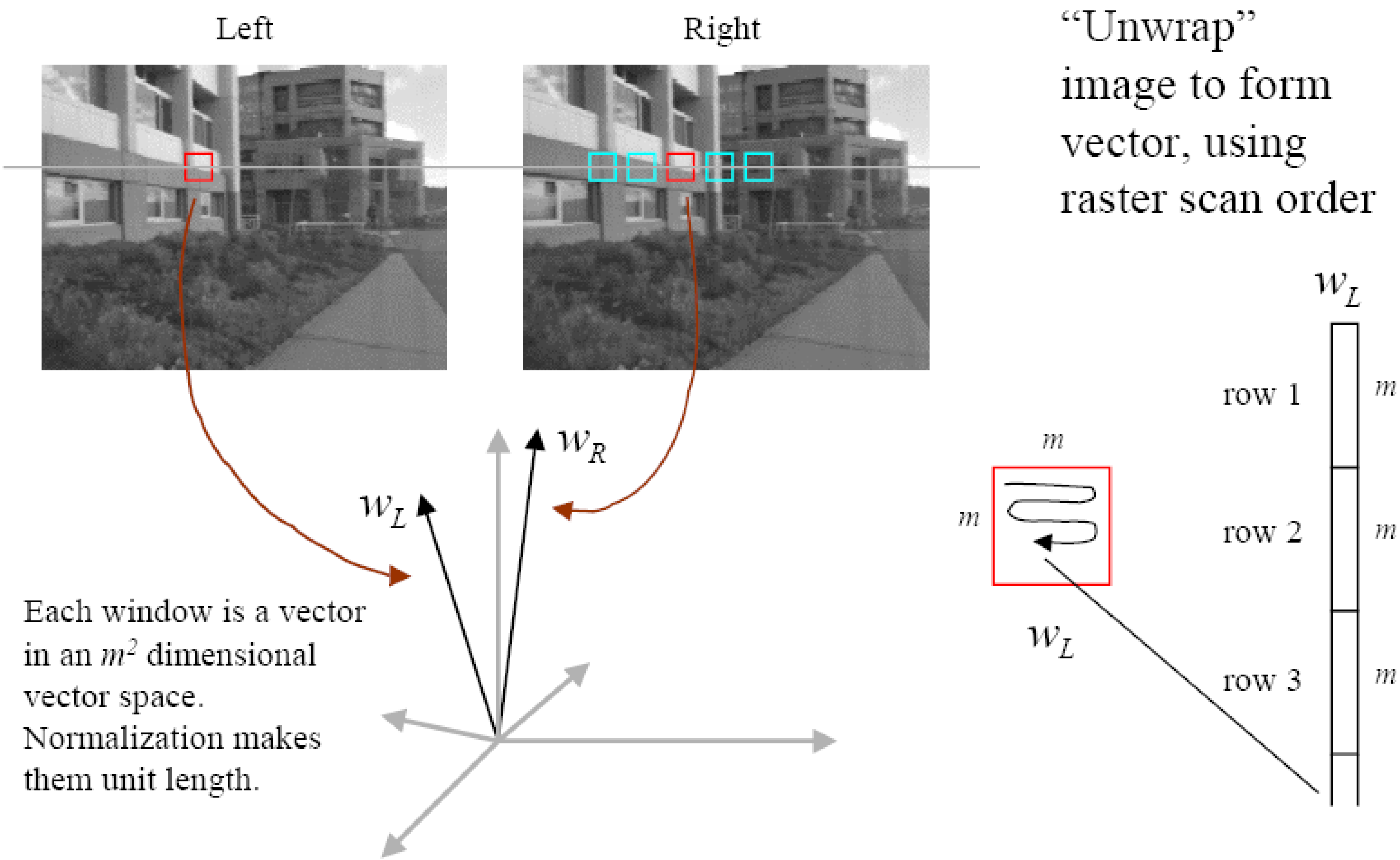
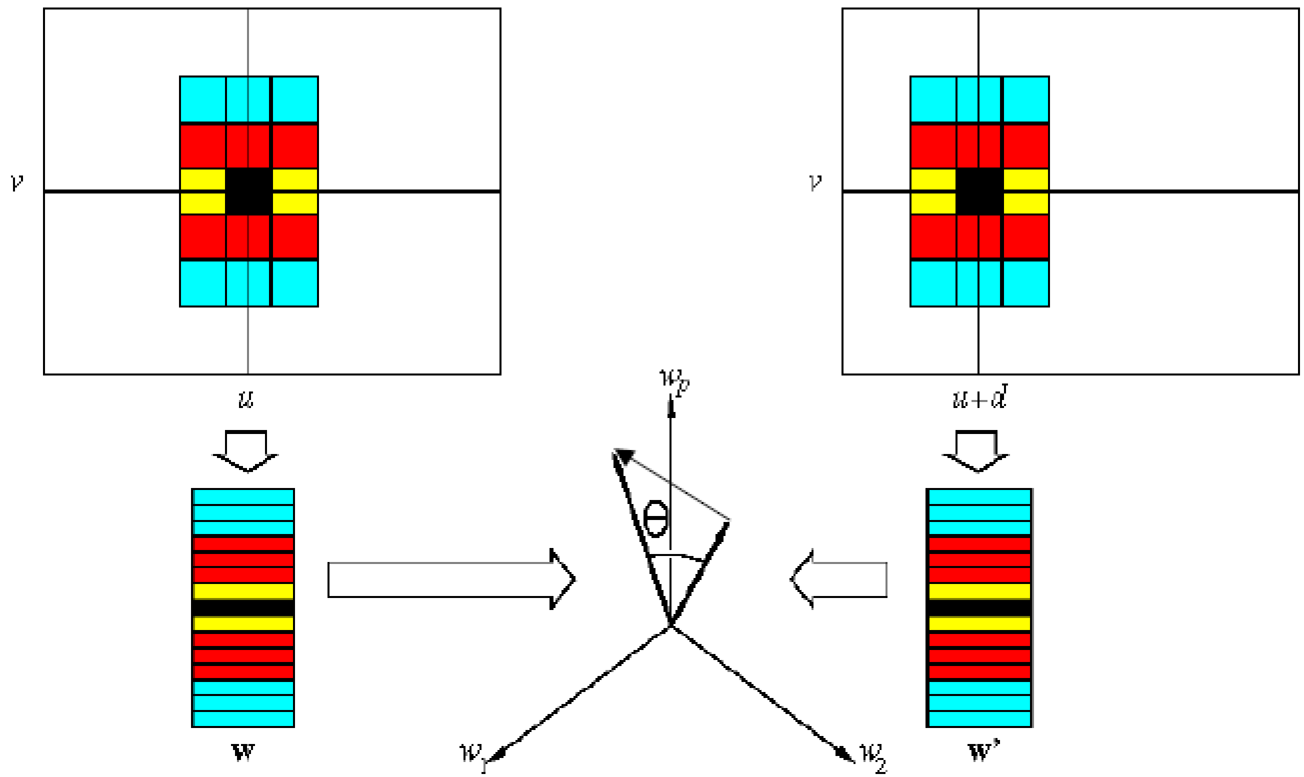
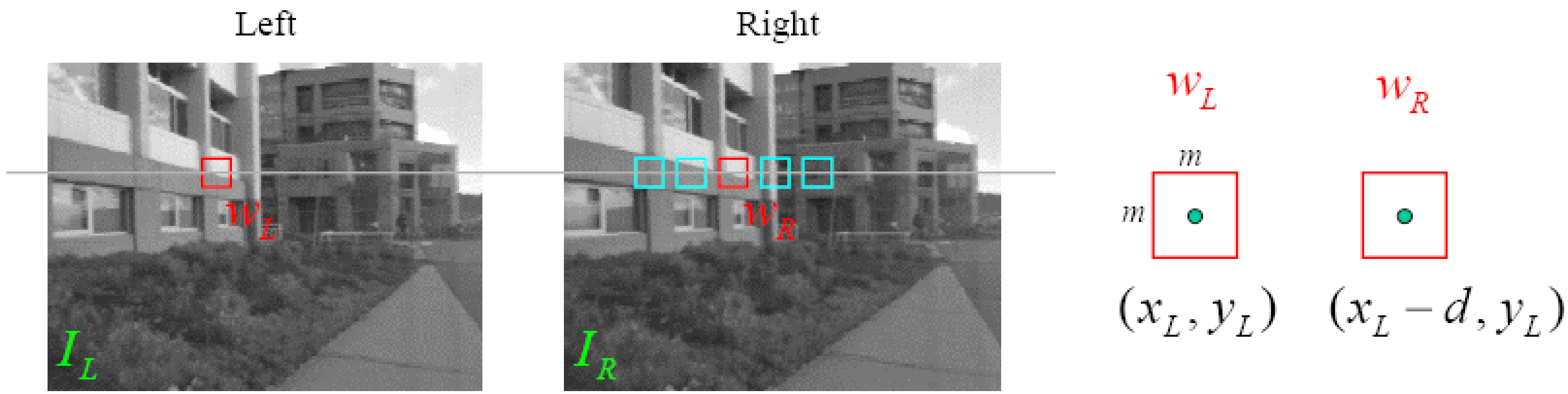


Image blocks as a vector



Sum of squared distance



w_L and w_R are corresponding m by m windows of pixels.

We define the window function :

$$W_m(x,y) = \{u, v \mid x - \tfrac{m}{2} \le u \le x + \tfrac{m}{2}, y - \tfrac{m}{2} \le v \le y + \tfrac{m}{2}\}$$

The SSD cost measures the intensity difference as a function of disparity :

$$C_r(x,y,d) = \sum_{(u,v)\in W_m(x,y)} [I_L(u,v) - I_R(u-d,v)]^2$$