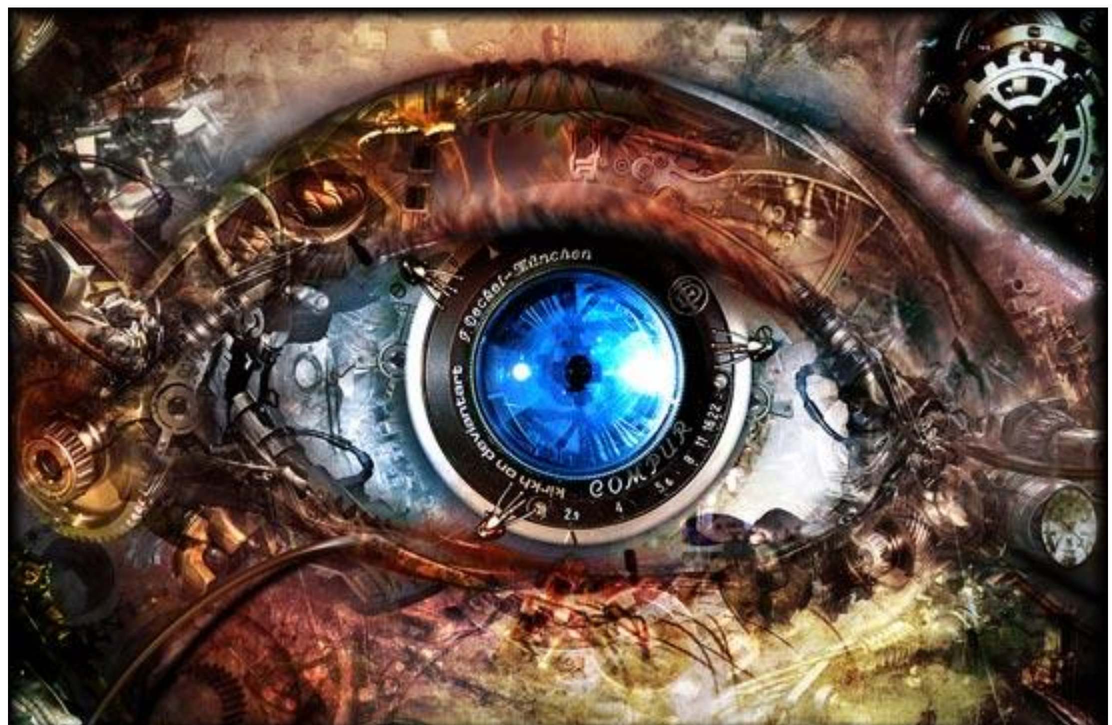
Chapter 6

Linear Filters

Prof. Fei Fei Li, Stanford University

Contents

- **2D Filter**
  - **Convolution**
  - **Linear Systems**



Multiplication

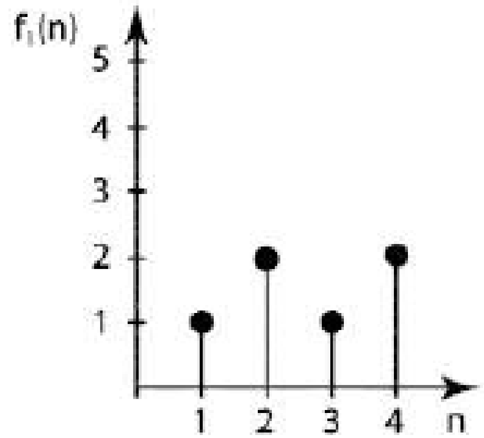
- *Using a simpler operation to generate a higher order operation*
- *Multiple summations*
- *General formula*

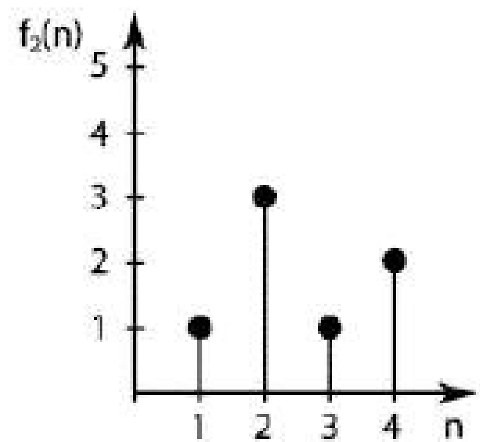
$$3 \cdot 7 = 7 + 7 + 7 = \sum_{1}^{3} 7 = 21$$

$$x \cdot y = \sum_{k=1}^{x} y$$

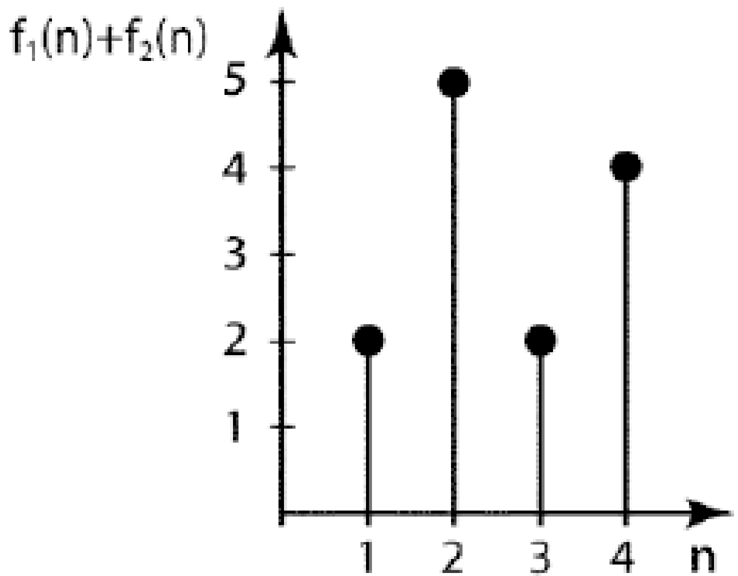
Doing the same with functions

- ***Applying the operations on each value separated.***
- ***The result is a function.***

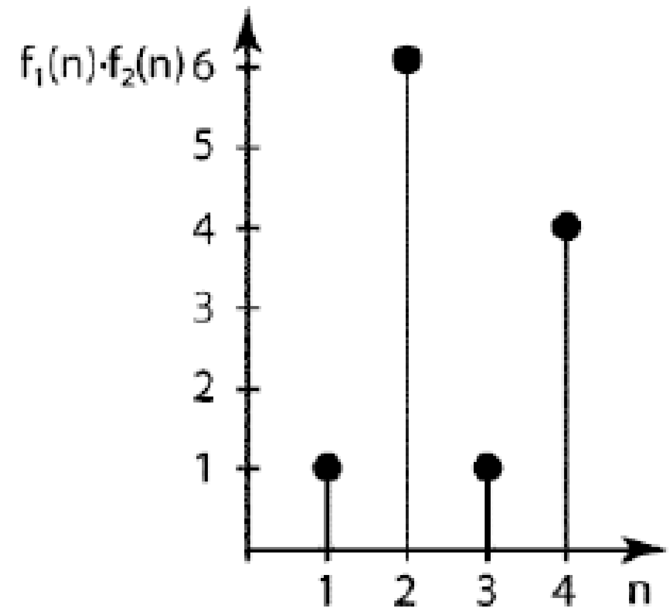




Multiplication and addition of two functions



$f_1 + f_2$



$f_1 \bullet f_2$

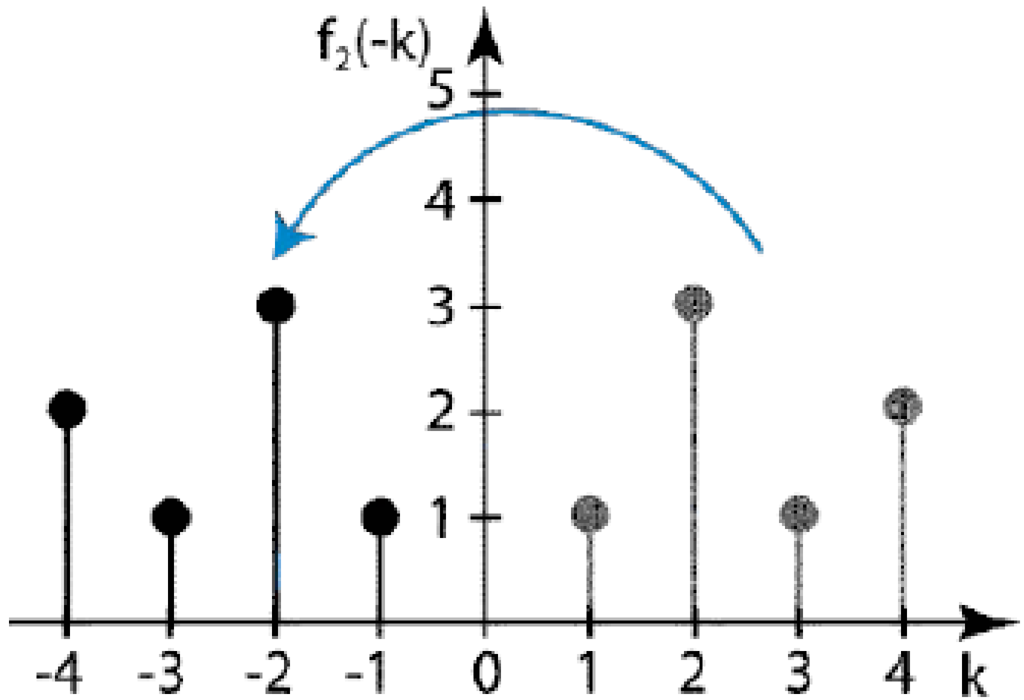
Convolution

- ***Operation that uses addition and multiplication.***
- ***Result is a function.***
- ***It is a way to combine to functions.***
- ***It is like weighting one function with the other.***
- ***Flipping one function and then summing up the products for each positions for a given offset n.***

$$g_{con}(n) = \sum_{k=-\infty}^{\infty} f_1(k) \cdot f_2(\mathrm{n} - k)$$

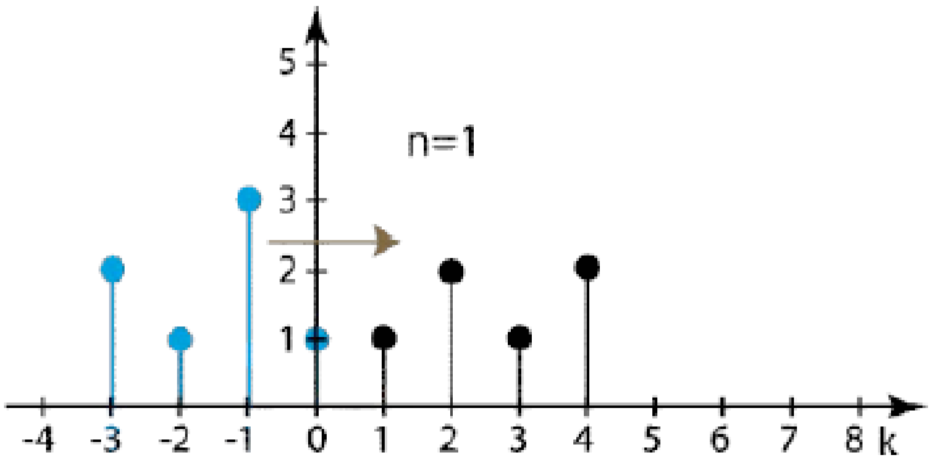
Discrete Convolution

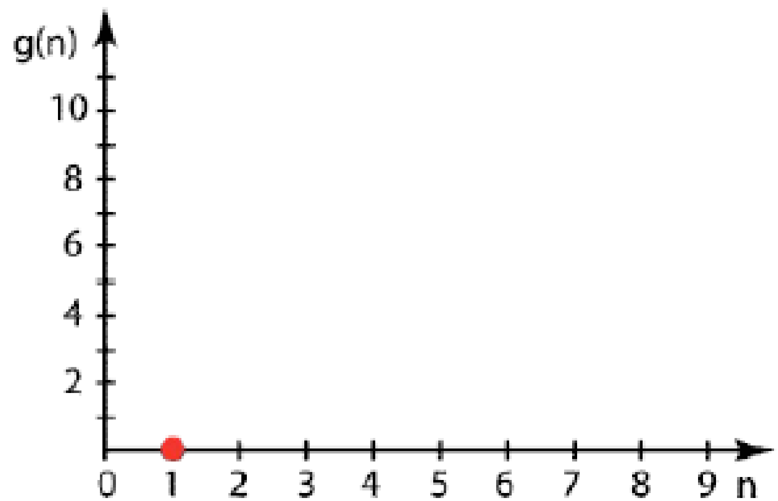
Flipping the function



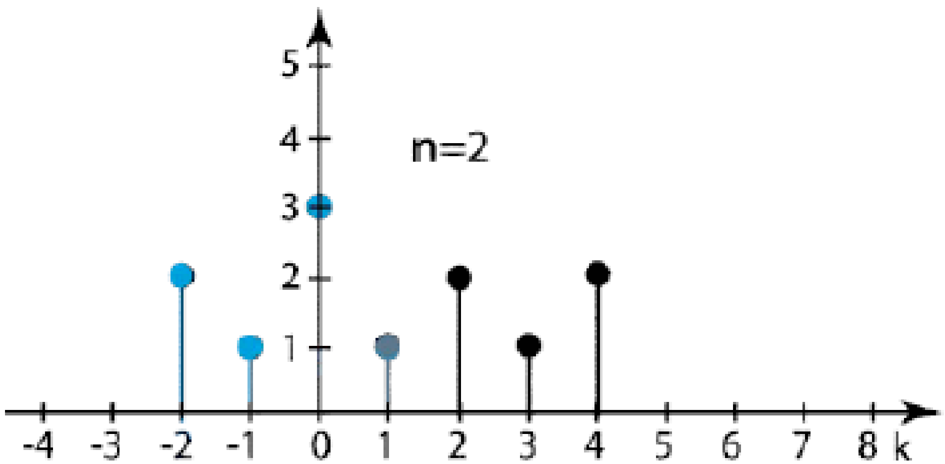
$f_2(-k)$

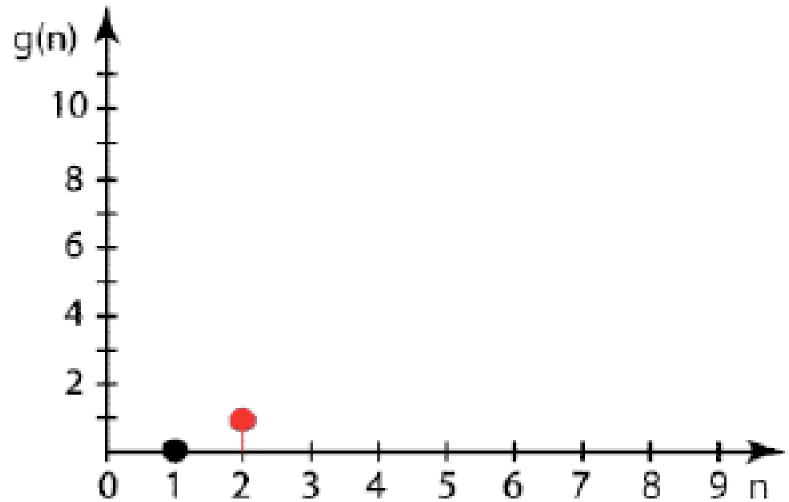
Multiply and add





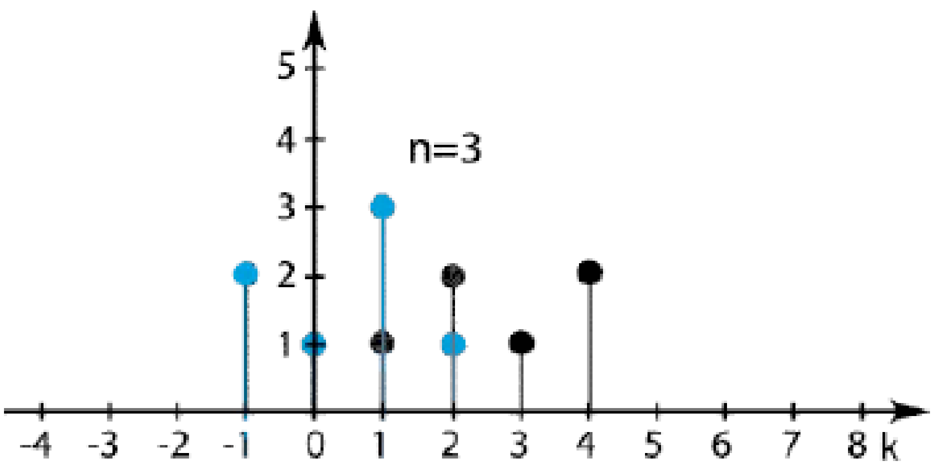
Multiply and add

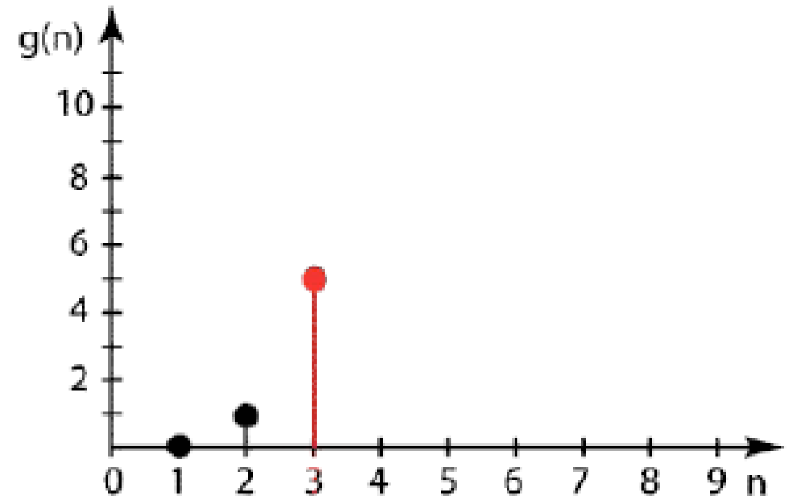




$$1 \bullet 1 = 1$$

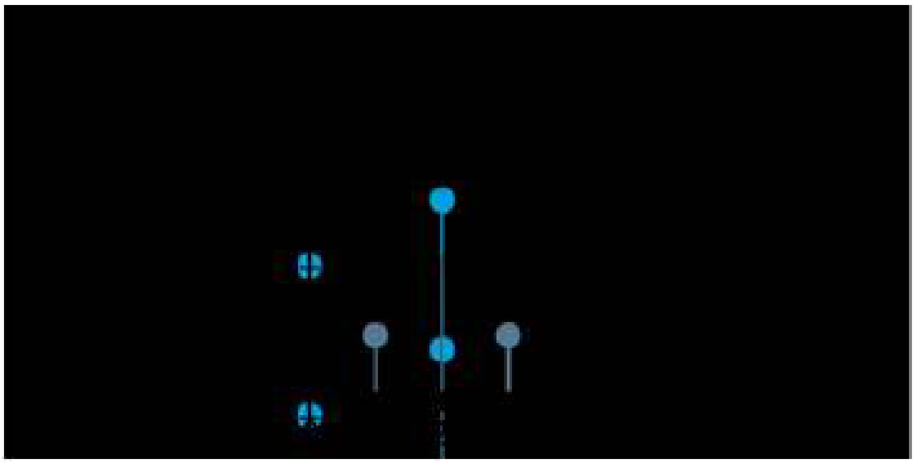
Multiply and add

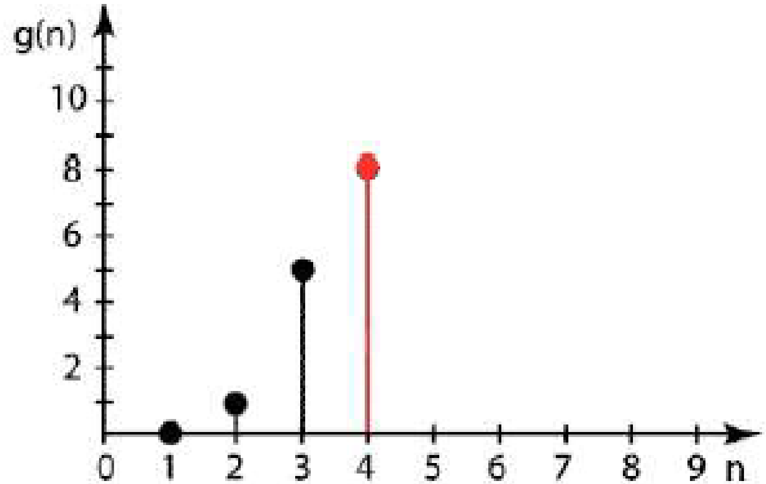




$$1 \bullet 3 + 2 \bullet 1 = 5$$

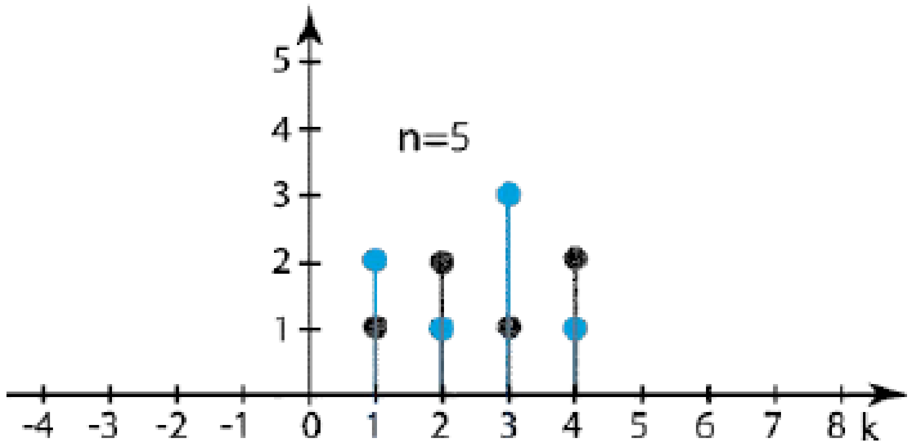
Multiply and add

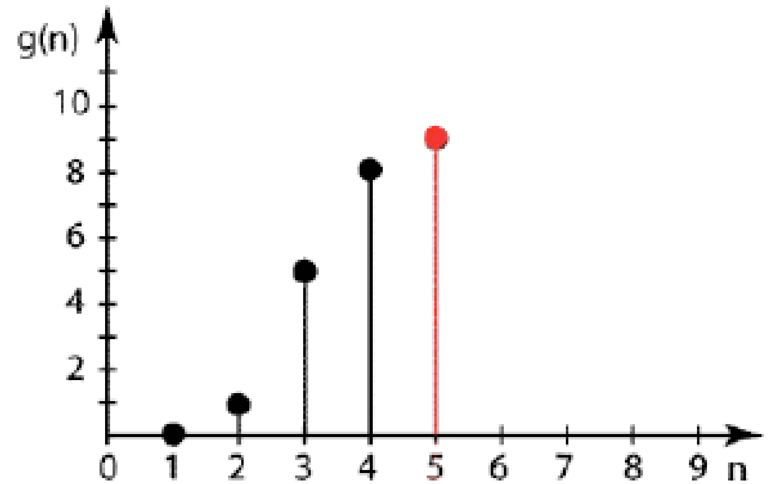




$$1 \bullet 1 + 2 \bullet 3 + 1 \bullet 1 = 8$$

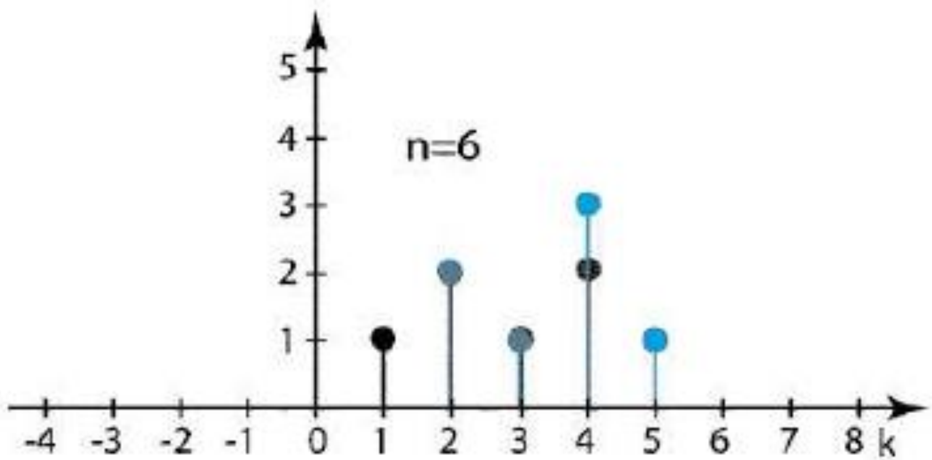
Multiply and add

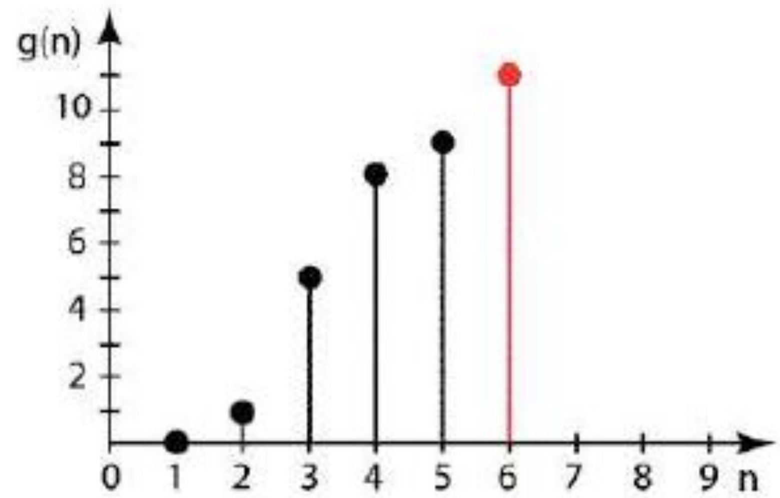




$$1\bullet2 + 2\bullet1 + 1\bullet3 + 2\bullet1 = 9$$

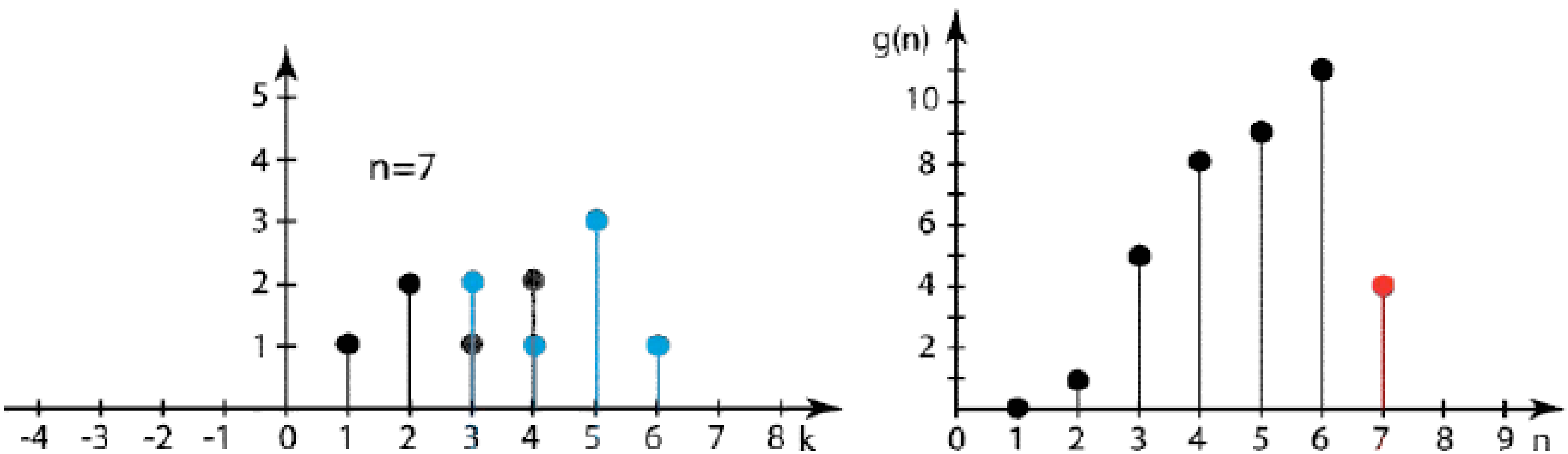
Multiply and add





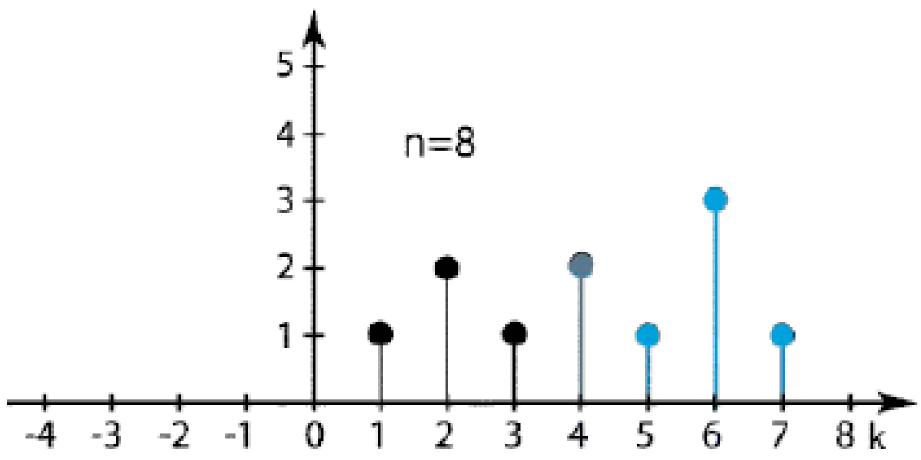
$$2 \bullet 2 + 1 \bullet 1 + 2 \bullet 3 = 11$$

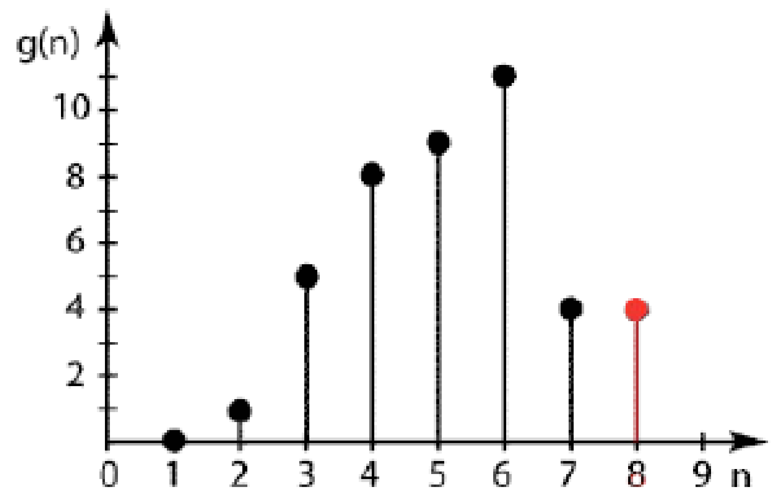
Multiply and add



$$1 \bullet 2 + 2 \bullet 1 = 4$$

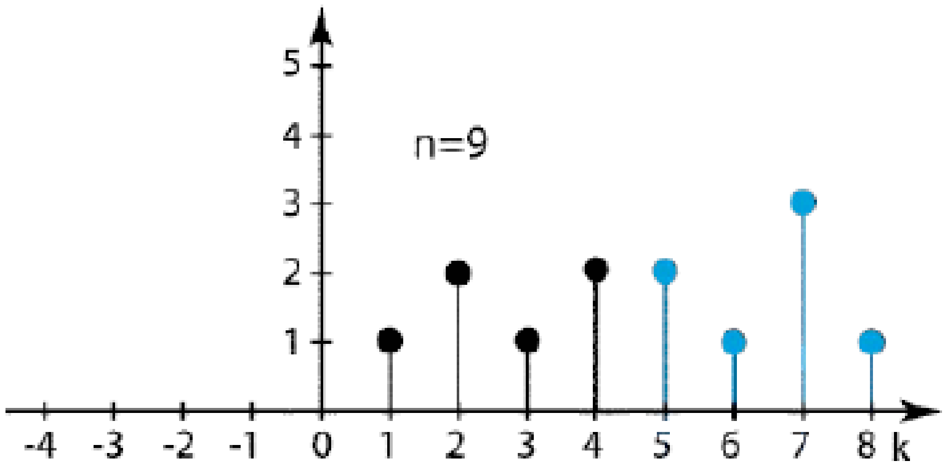
Multiply and add

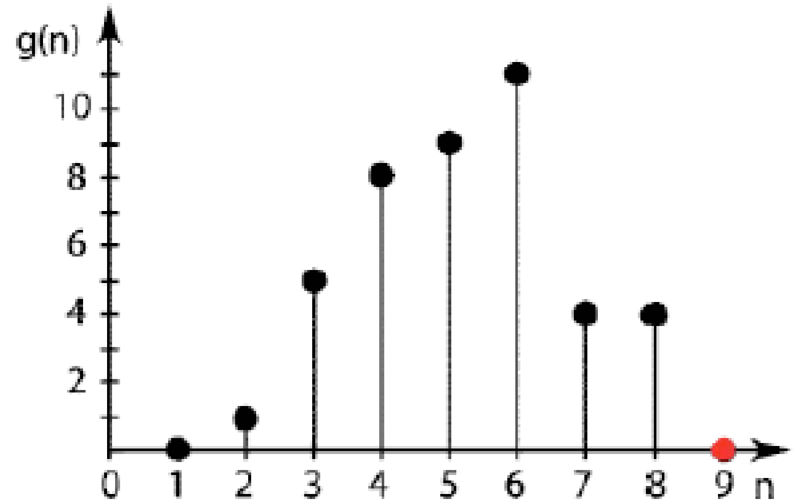




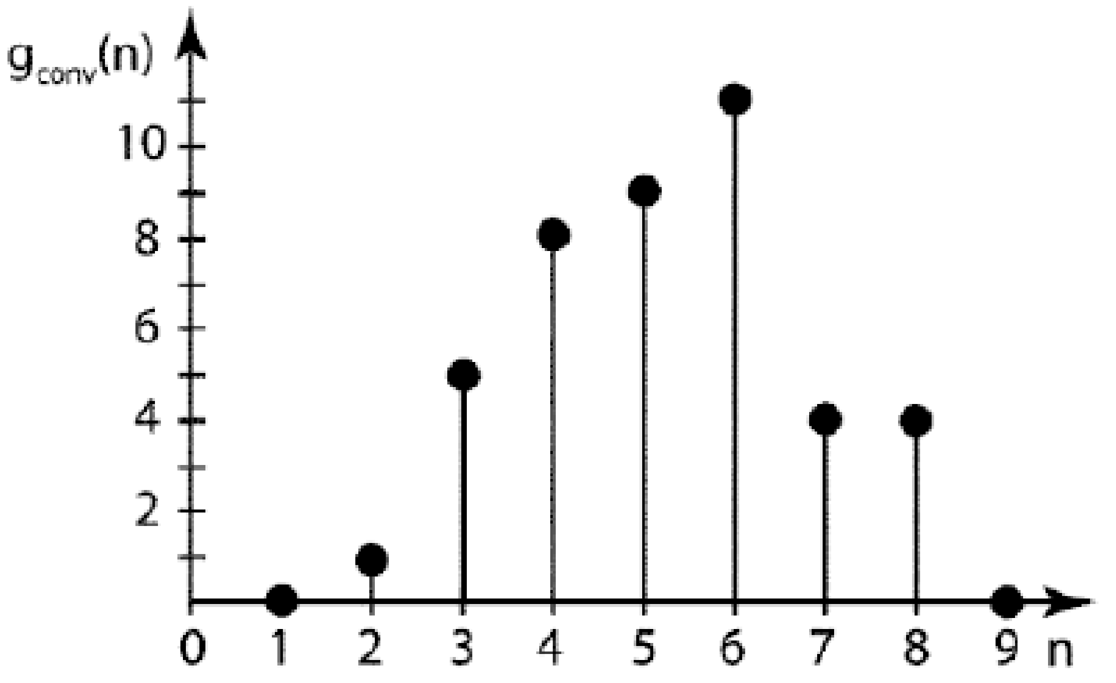
$$2 \bullet 2 = 4$$

Multiply and add





Result

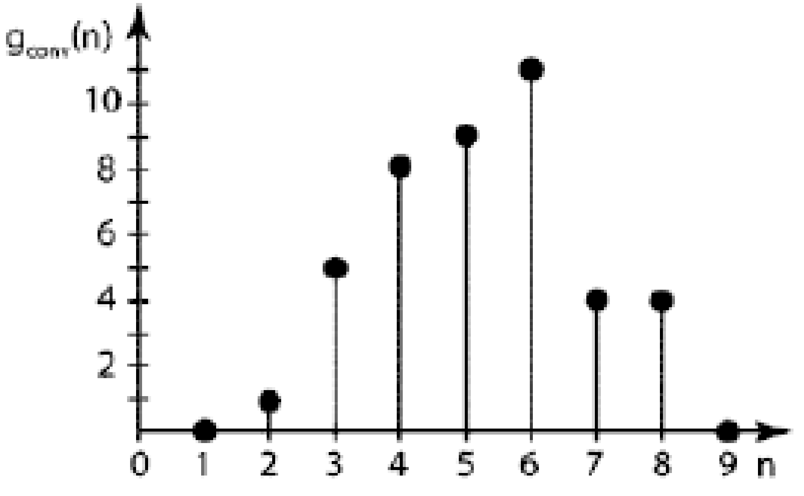


$$\text{final length} = \text{length}(f_1(n)) + \text{length}(f_2(n)) - 1$$

Comparison

Convolution

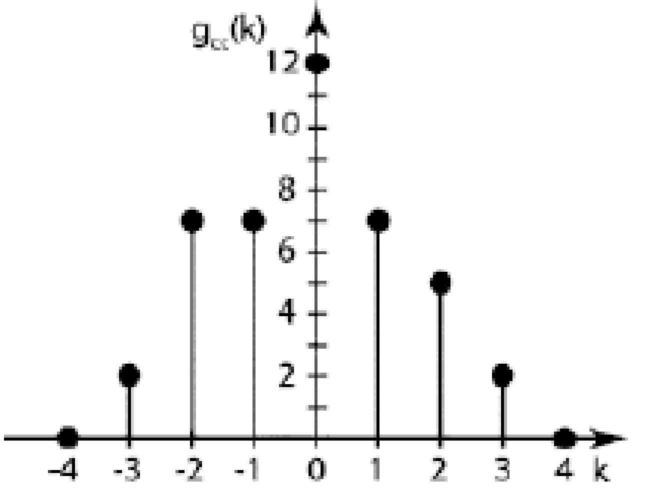
$$g_{con}(n)=\sum_{k=-\infty}^{\infty} f_1(k)\cdot f_2(n-k)$$



Crosscorrelation

$$g_{cc}(k)=\sum_{n=-\infty}^{\infty} f_1(n)\cdot f_2(n+k)$$

$$g_{cc}(n)=\sum_{k=-\infty}^{\infty} f_1(k)\cdot f_2(n+k)$$



Continuous Convolution

- ***Summation becomes an integral***

$$g_{con}(n)=\sum_{k=-\infty}^{\infty} f_1(k)\cdot f_2(n-k)$$

$$\Downarrow$$

$$g(t)=f_1(t)*f_2(t)=\int_{-\infty}^{\infty} f_1(\tau)\cdot f_2(t-\tau)\,d\tau$$

Properties

- **Commutative**

$$f(n)*g(n)=g(n)*f(n)$$

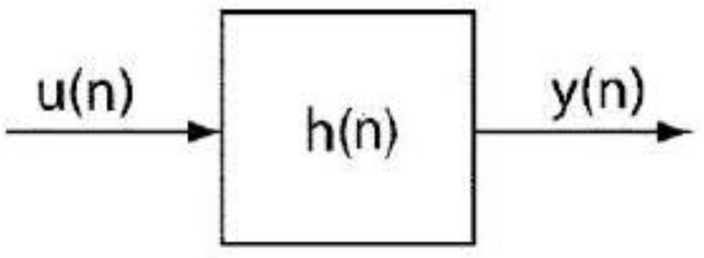
- **Associative**

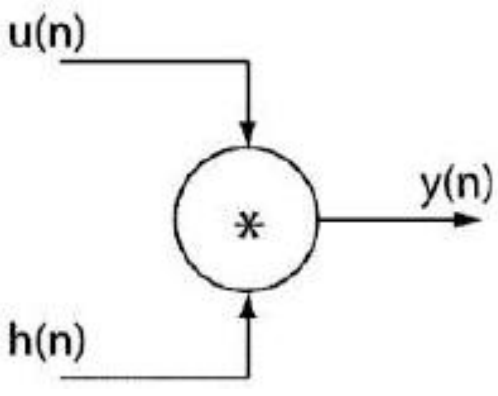
$$(f(n)*g(n))*h(\mathrm{n})=f(\mathrm{n})*(\mathrm{g}(\mathrm{n})*h(n))$$

- **Distributive**

$$f(n)*(g(n))+h(\mathrm{n}))=f(\mathrm{n})*\mathrm{g}(\mathrm{n})+f(\mathrm{n})*h(n)$$

Filter





- **Signal y(n) is a convolution of u(n) with the *Transfer function* h(n)**
- **Filtering can be done by the convolution of two signals**

2D Filtering

A 2D in1age f[i,j] can be filtered by a 2D kernel h[u,v] to produce an output image g[i,j]

$$g[i,j]=\sum_{u=-k}^{k}\sum_{v=-k}^{k}h[u,v]\cdot f[i+u,j+v]$$

This is called a cross-correlation operation and written:

$$g=h\circ f$$

h is called tl1e "filter", "kernel" or "mask".

2D Filtering

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] \cdot f[i-u, j-v]$$

It is written: $\boxed{g = h * f}$ $= \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[-u,-v] \cdot f[i+u, j+v]$

Suppose h is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

If $h[u,v] = h[-u,-v]$ then there is no difference between convolution and cross-correlation

Example finding edges

Differentiating to find the steep parts of the picture

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Example Simple Noise Reduction

- ***Cutting of the high frequency noise by low pass filtering with the sine like kernel***
- ***For not changing the aspect of the image the sum of the kernel must be 1***

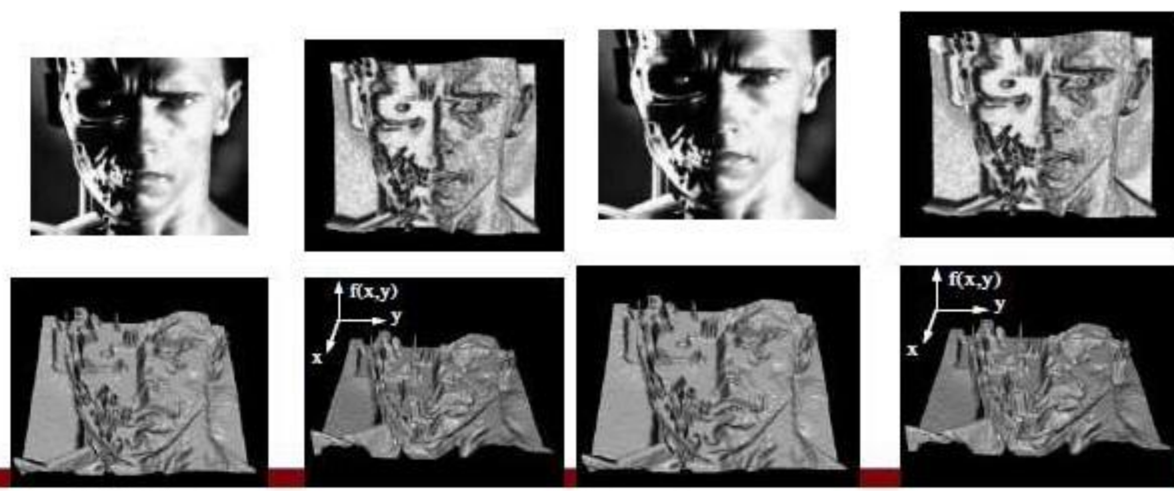
1/13

1	1	1
1	5	1
1	1	1

Images as functions

- ***An Image as a function f from R^2 to R^M:***
  - ***$f(x,y)$ gives the intensity at position (x, y)***
  - ***Defined over a rectangle, with a finite range:***

$$f:\underbrace{[a,b]\mathrm{x}[c,d]}_{\text{Domanin Support}}\rightarrow\underbrace{[0,255]}_{\text{range}}$$



Images as functions

- ***An Image as a function f from R^2 to R^M:***
  - ***$f(x,y)$ gives the intensity at position (x, y)***
  - ***Defined over a rectangle, with a finite range:***

$$f : \underbrace{[a,b]\mathrm{x}[c,d]}_{\text{Domanin Support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$

- ***A color image:***

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Images as discrete functions

- ***Images are usually digital (discrete):***
  - ***Sample the 20 space on a regular grid***
- ***Represented as a matrix of integer values***

pixel

j

i

62	79	23	119	120	05	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Images as discrete functions

Cartesian coordinates

$$f[n,m] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,1] & f[0,1] & f[1,1] & \\ \dots & f[-1,0] & \underline{f[0,0]} & f[1,0] & \dots \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

Notation for discrete functions

Images as discrete functions

Array coordinates

$$A = \begin{vmatrix} a_{11} & . & . & . & a_{1M} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ a_{N1} & & & & a_{NM} \end{vmatrix}$$

Matlab notation

$$A = \begin{vmatrix} a_{00} & . & . & . & a_{0M-1} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ a_{N-10} & & & & a_{N-1M-1} \end{vmatrix}$$

C++ notation

Systems and Filters

- ***Filtering:***
  - *Form a **new image** whose pixels are a combination original pixel values*

Goals:

- ***Extract useful information from the images***
  - *Features (edges, corners, blobs…)*

- ***Modify or enhance image properties:***
  - *Super-resolution; in-painting; de-noising*

De-noising

Original

Salt and pepper noise

Super-resolution

In-painting

Bertamio et al

2D discrete-space systems (filters)

$$f[n,m] \longrightarrow \boxed{\text{System } \mathcal{S}} \longrightarrow g[n,m]$$

$$g = \mathcal{S}[f], \quad g[n,m] = \mathcal{S}\{f[n,m]\}$$

$$f[n,m] \xrightarrow{\mathcal{S}} g[n,m]$$

Filters: Examples

- 2D DS moving average over a 3 × 3 window of neighborhood

$$g[n,m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k,l]$$

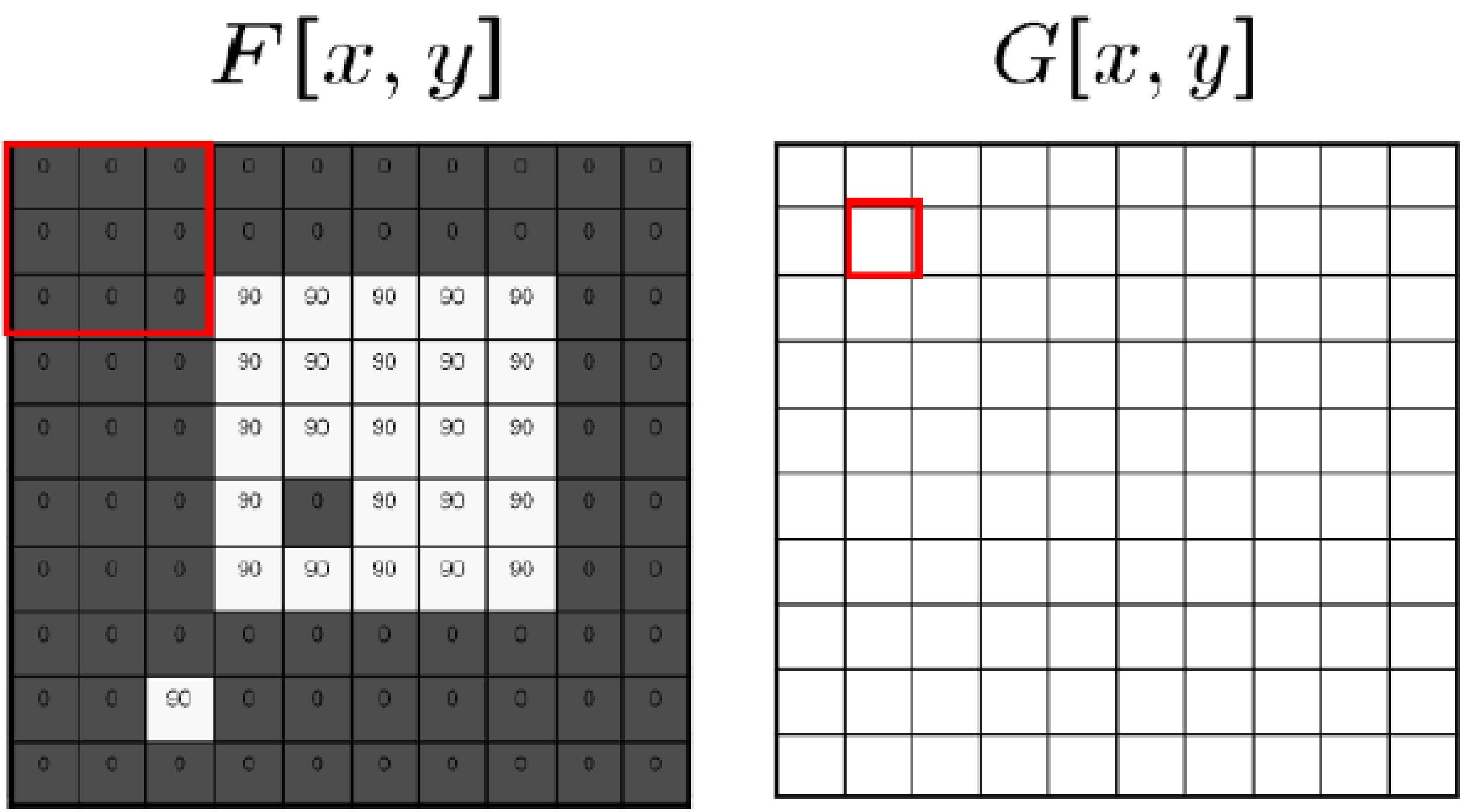
$$= \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n-k, m-l]$$

h ***Filter***

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

convolution $$(f*h)[m,n] = \frac{1}{9}\sum_{k,l} f[k,l]\, h[m-k, n-l]$$

Moving average



Moving average

$F[x,y]$

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

$G[x,y]$

	0	10							

$$(f * g)[m,n] = \sum_{k,l} f[k,l]\, g[m-k, n-l]$$

Moving average

$F[x, y]$

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

$G[x, y]$

	0	10	20						

$$(f * g)[m, n] = \sum_{k,l} f[k, l]\, g[m-k, n-l]$$

Moving average

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

$$(f * g)[m,n] = \sum_{k,l} f[k,l]\, g[m-k, n-l]$$

Moving average

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

$$(f * g)[m,n] = \sum_{k,l} f[k,l]\, g[m-k, n-l]$$

Moving average

$F[x, y]$

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

$G[x, y]$

	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	10	10	0	0	0	0	0	

$$(f * g)[m,n] = \sum_{k,l} f[k,l]\, g[m-k, n-l]$$

Moving average

In summary:

- ***Replaces each pixel with an average of its neighborhood.***
- ***Achieve smoothing effect (remove sharp features)***

$g[\cdot,\cdot]$

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

Moving average

Shift-invariance

- If $f[n,m] \xrightarrow{S} g[n,m]$ then

$$\boxed{f[n-n_0, m-m_0] \xrightarrow{S} g[n-n_0, m-m_0]}$$

for every input image f[n,m] and shifts n_0,m_0

- Is the moving average shift invariant a system ?

Is the moving average system is shift invariant?

$F[x, y]$

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

$G[x, y]$

	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	10	10	0	0	0	0	0	

Is the moving average system is shift invariant?

$$f[n,m] \xrightarrow{\mathcal{S}} g[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k, m-l]$$

$$f[n-n_0, m-m_0]$$

$$\xrightarrow{\mathcal{S}} g[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k, m-l]$$

$$= \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[(n-n_0)-k, (m-m_0)-l]$$

$$= g[n-n_0, m-m_0]$$ Yes!

Linear Systems (filters)

$$f(x,y) \rightarrow \boxed{\mathcal{S}} \rightarrow g(x,y)$$

- ***Linear filtering:***
  - ***Form a new image whose pixels are a weighted sum of original pixel values***
  - ***Use the same set of weights at each point***
- ***S is a linear system (function) iff it S satisfies***

$$\mathcal{S}[\alpha f_1 + \beta f_2] = \alpha\mathcal{S}[f_1] + \beta\mathcal{S}[f_2]$$

superposition property

Linear Systems (filters)

$$f(x,y) \rightarrow \boxed{S} \rightarrow g(x,y)$$

- Is the moving average a system linear?

$$f[n,m] \xrightarrow{S} g[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k, m-l]$$

$$S(\alpha f) = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} \alpha f[n-k, m-l]$$

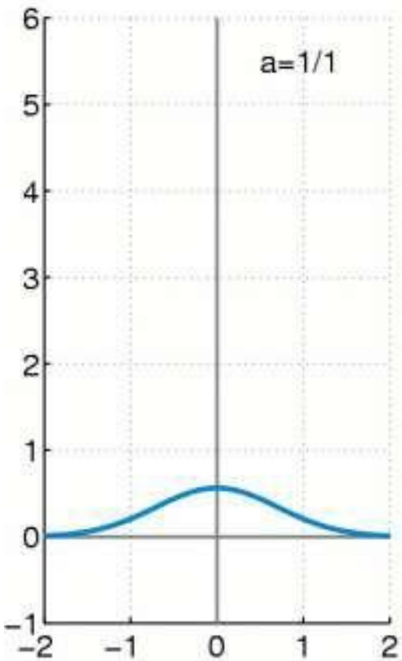
$$= \alpha\{\frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k, m-l]\}$$

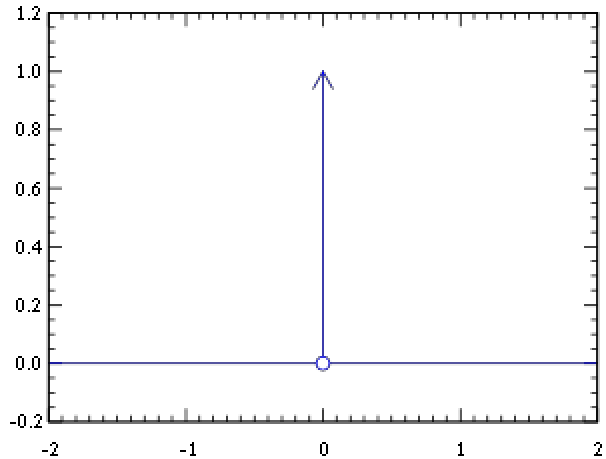
$$= \alpha S(f)$$

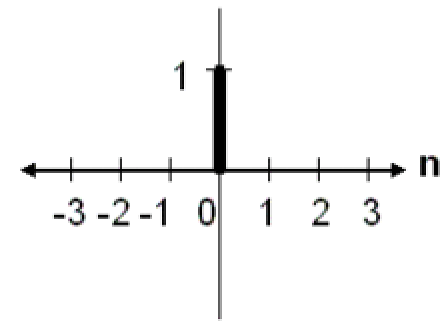
- Is thresholding a system linear?
  - f1[n,m] + f2[n,m] > T
  - f1[n,m] < T
  - f2[n,m]<T

No!

Linear Shift Invariant System - LSI







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

The Dirac delta function as the limit (in the sense of distributions) of the sequence of Gaussians

Kronecker delta

$$\delta_a(x) = \frac{1}{a\sqrt{\pi}} \mathrm{e}^{-x^2/a^2}$$

$$a \to 0.$$

The impulse can be modeled as a Dirac delta function for continuous-time systems, or as the Kronecker delta for discrete-time systems.

LSI (linear shift invariant) Systems

Impulse response

$$\delta_2[n,m] \rightarrow \boxed{\mathcal{S}} \rightarrow h[n,m]$$

$$\delta_2[n-k,m-l] \rightarrow \boxed{\mathcal{S}\ (\mathrm{SI})} \rightarrow h[n-k,m-l]$$

$$\delta[x-a,y-b] = \begin{cases} 1 & x=a \wedge y=b \\ 0 & else \end{cases}$$

LSI (linear shift invariant) Systems

Example: impulse response of the 3 by 3 moving average filter:

$$h[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1}\delta_2[n-k,m-l]$$

$$= \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

h

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

LSI (linear shift invariant) Systems

An LSI system is completely specified by its impulse response.

sifting property of the delta function

$$f[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\,\delta_2[n-k, m-l]$$

superposition

$$\rightarrow \boxed{\mathcal{S}\ \text{LSI}} \rightarrow \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\,h[n-k, m-l]$$

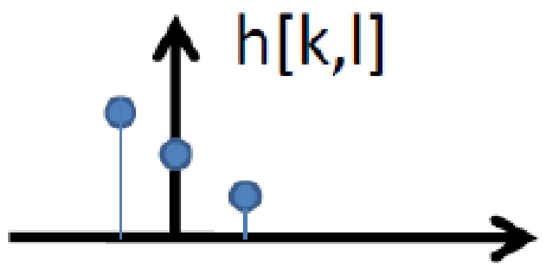
$$\delta_2[n,m] \rightarrow \boxed{\mathcal{S}} \rightarrow h[n,m]$$

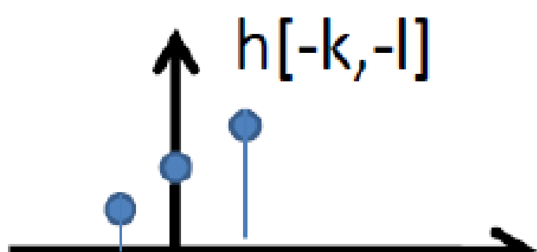
Discrete convolution

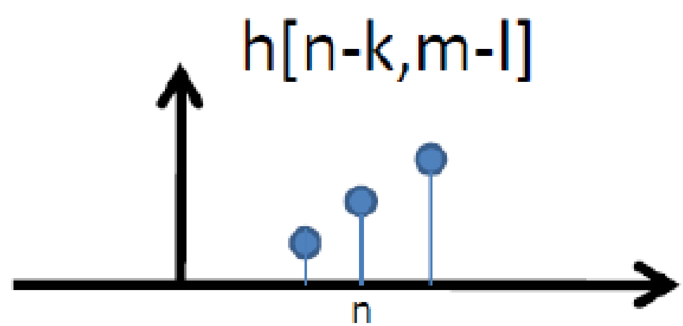
$$= f[n,m] ** h[n,m]$$

Discrete convolution

- ***Fold h[n,m] about origin to form h[-k,-l]***
- ***Shift the folded results by n,m to form h[n - k,m - l]***
- ***Multiply h[n - k,m - l] by f[k,l]***
- ***Sum over all k,l***
- ***Repeat for every n, m***

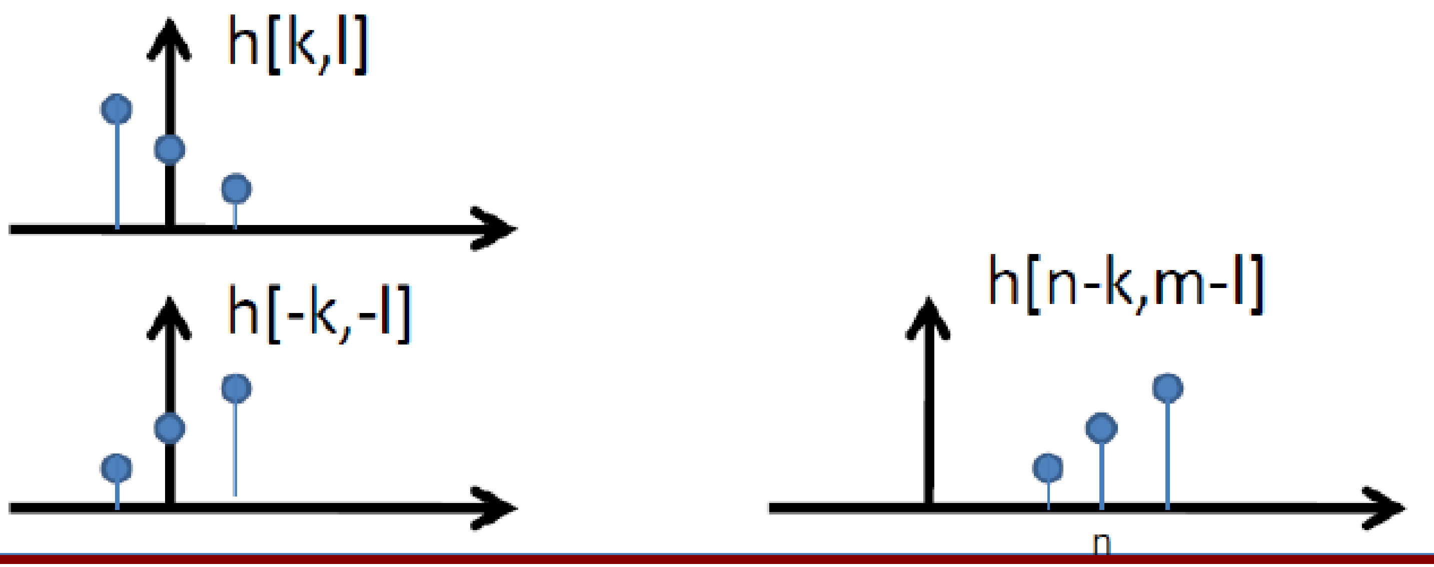






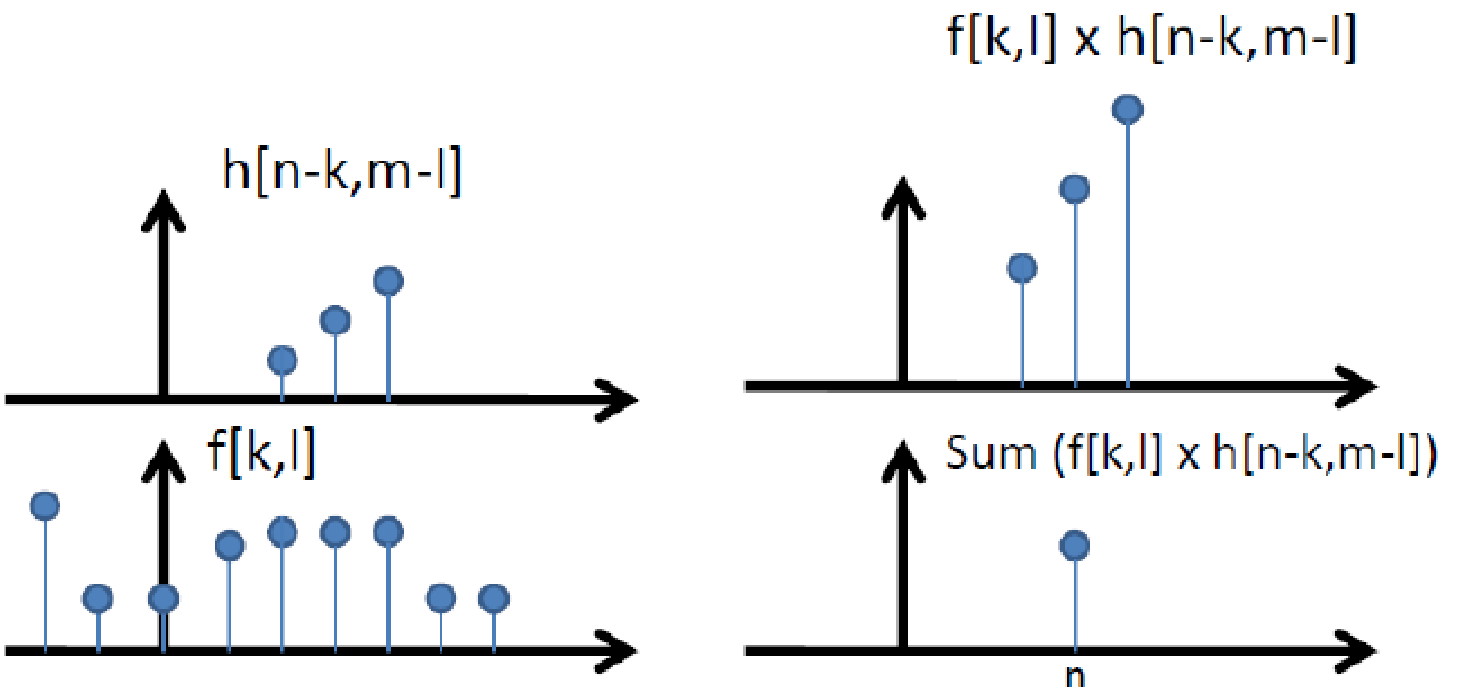
Discrete convolution

- ***Fold h[n,m] about or igin to form h[-k,-l]***
- ***Shift the folded results by n,m to form h[n - k,m - l]***
- ***Multiply h[n - k,m - l] by f[k,l]***
- ***Sum over all k,l***
- ***Repeat for every n, m***



Discrete convolution

- ***Fold h[n,m] about or igin to form h[-k,-l]***
- ***Shift the folded results by n,m to form h[n - k,m - l]***
- ***Multiply h[n - k,m - l] by f[k,l]***
- ***Sum over all k,l***
- ***Repeat for every n, m***



Convolution in 2D - Examples

Original

•0	•0	•0
•0	•1	•0
•0	•0	•0

=

?

Convolution in 2D - Examples

Original

•0	•0	•0
•0	•1	•0
•0	•0	•0

=

Filtered
(no change)

Convolution in 2D - Examples

Original

•0	•0	•0
•0	•0	•1
•0	•0	•0

=

?

Convolution in 2D - Examples

Original

•0	•0	•0
•0	•0	•1
•0	•0	•0

=

Shifted left
By 1 pixel

Convolution in 2D - Examples

Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} = \; ?$$

Convolution in 2D - Examples

Original

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

Blur (with a box filter)

Convolution in 2D - Examples

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

(Note that filter sums to 1)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \overbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}}^{\text{details}}$$

Convolution in 2D – Sharpening Filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Original

Sharpening filter: ***Accentuates differences with local average***

Convolution properties

- **Commutative property:**

$$f ** h = h ** f$$

- **Associative property:**

$$(f ** h_1) ** h_2 = f ** (h_1 ** h_2)$$

- **Distributive property:**

$$f ** (h_1 + h_2) = (f ** h_1) + (f ** h_2)$$

The order doesn't matter! $h_1 ** h_2 = h_2 ** h_1$

Convolution properties

- **Shift property:**

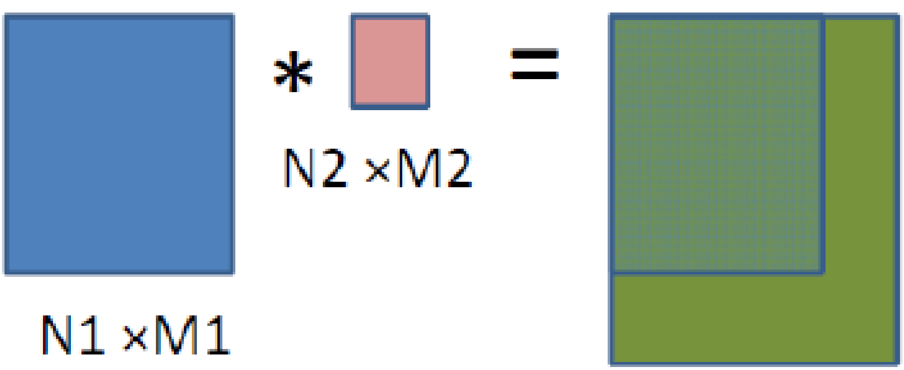
$$f[n,m] ** \delta_2[n-n_0, m-m_0] = f[n-n_0, m-m_0]$$

- **Shift-invariance:**

$$\begin{aligned} g[n,m] &= f[n,m] ** h[n,m] \\ &\Longrightarrow f[n-l_1, m-l_1] ** h[n-l_2, m-l_2] \\ &= g[n-l_1-l_2, m-l_1-l_2] \end{aligned}$$

Image support and edge effect

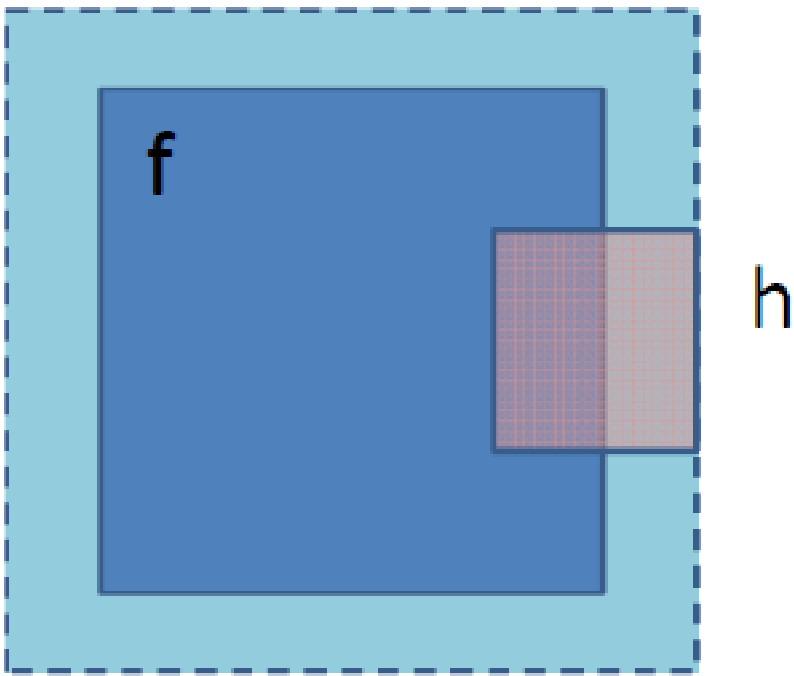
- *A computer will only convolve finite support signals.*
  - *That is: images that are zero for n,m outside some recta ngular region*
- *MATLAB's conv2 performs 2D DS convolution of finite support signals.*



$$(N1 + N2 - 1) \times (M1 + M2 - 1)$$

Image support and edge effect

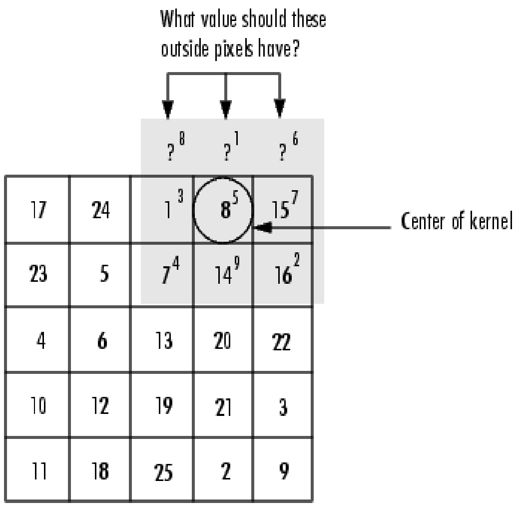
- ***A computer will only convolve finite support signals.***
- ***What happens at the edge?***



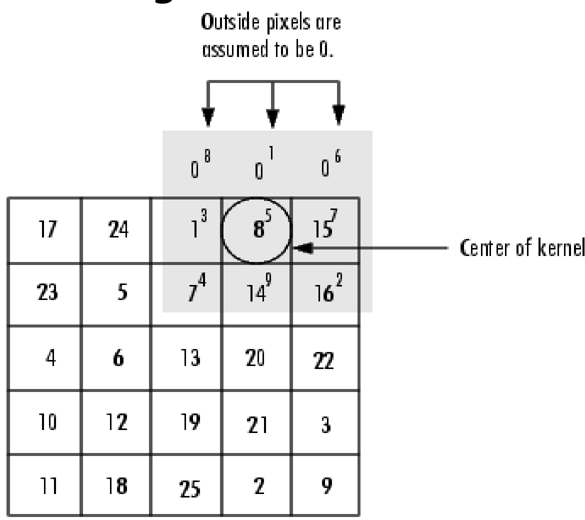
- zero “padding”
- edge value replication
- mirror extension
- more (beyond the scope of this class)

-> Matlab conv2 uses zero-padding

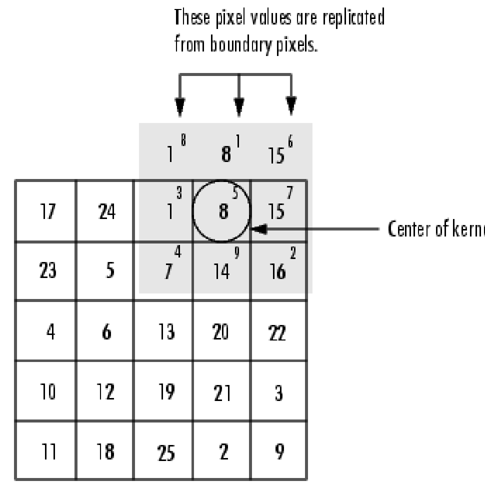
Boundary Padding Options



Zero-Padding



Replicated Boundary Pixels



See the reference page for imfilter for details.

Cross correlation

Cross correlation of two 2D signals f[n,m] and g[n,m]

$$r_{fg}[k,l] \triangleq \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n,m]\, g^*[n-k,m-l]$$

$$= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n+k,m+l]\, g^*[n,m], \quad k,l \in \mathbb{Z}.$$

(k, l) is called the **lag**

- Equivalent to a convolution without the flip

$$r_{fg}[n,m] = f[n,m] ** g^*[-n,-m]$$

Convolution vs. Correlation

- A **convolution** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a filtering operation

Matlab:
`conv2`

- **Correlation** compares the ***similarity** of **two** sets of* ***data.*** Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best .
  - correlation is a measure of relatedness of two signals

Matlab:
`filter2`
`imfilter`

Filtering: Boundary Issues

- ***What is the size of the output?***
- ***MATLAB:*** `filter2(g,f,shape)`
  - ***shape = 'full': output size is sum of sizes of f and g***
  - ***shape = 'same': output size is same as f***
  - ***shape = 'valid': output size is difference of sizes of f and g***