Imperial College London

Image Filtering II

Dr Wenjia Bai

Department of Computing & Brain Sciences

How to describe filtering mathematically?

- Filtering in signal processing
- Filtering described as convolution

Recap on image filtering

1/9	1/9	1/9	111	110	123	130	130
1/9	1/9	1/9	111	113	120	126	125
1/9	1/9	1/9	108	113	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

	147						

Recap on image filtering

199	1/9	1/9	1/9	110	123	130	130
189	1/9	1/9	1/9	113	120	126	125
130	1/9	1/9	1/9	113	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

	147	126					

Recap on image filtering

199	192	1/9	1/9	1/9	123	130	130
189	149	1/9	1/9	1/9	120	126	125
130	100	1/9	1/9	1/9	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

	147	126	114				

Recap on image filtering

199	192	158	111	110	123	130	130
189	149	108	111	113	120	126	125
130	100	98	108	113	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	1/9	1/9	1/9
77	80	72	71	60	1/9	1/9	1/9
68	67	63	58	53	1/9	1/9	1/9

	147	126	114	114	118	122	
	117	108	107	111	113	113	
	99	99	102	106	107	105	
	91	94	93	93	94	94	
	85	86	81	78	78	79	
	76	74	68	62	62	64	

How to describe filtering mathematically?

- For simplicity, we start from 1D case.

kernel

1/3	1/3	1/3

189	149	108	111	113	120	126	125

input: f

		123					

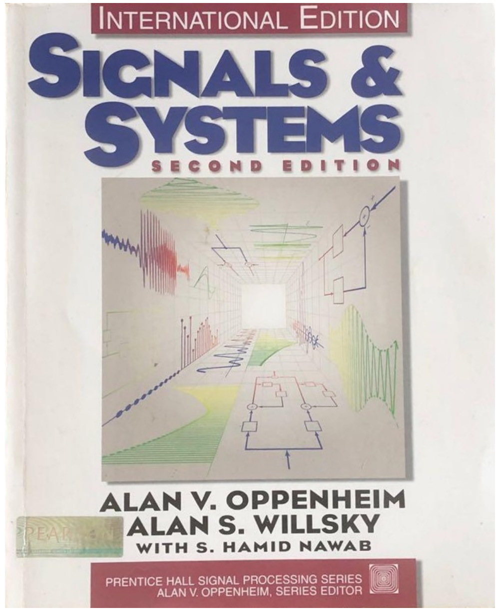
output: g

- It can be written as a weighted average in a window,

$$g[n] = \frac{1}{3} \cdot f[n-1] + \frac{1}{3} \cdot f[n] + \frac{1}{3} \cdot f[n+1]$$

Filtering in signal processing

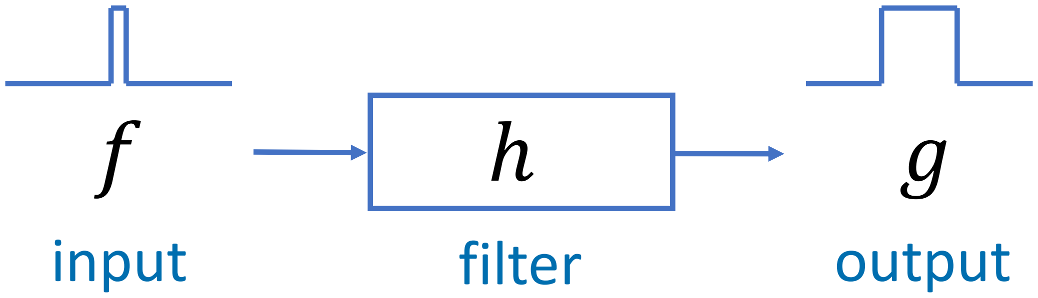
- We have an input signal f, process it and generate output signal g.
- This is called filtering, intensively studied in signal processing.
- A filter (or filtering) is a device (or process) that removes unwanted components or features from a signal.
- In other words, it keeps or enhances wanted features.



Signal processing

Filter

- A filter h takes an input signal f, processes it and generates output signal g.



Filter

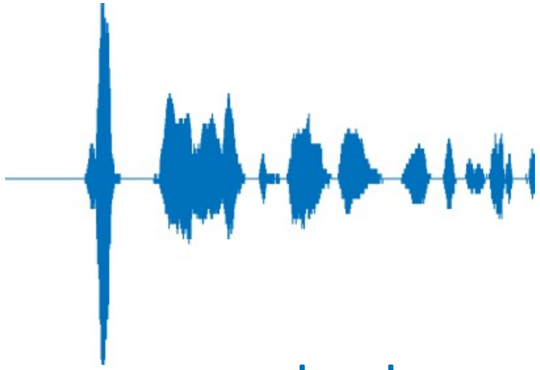
- A filter can be an electrical hardware device.

Filter

- A filter can also be a digital software.

input

filter



output

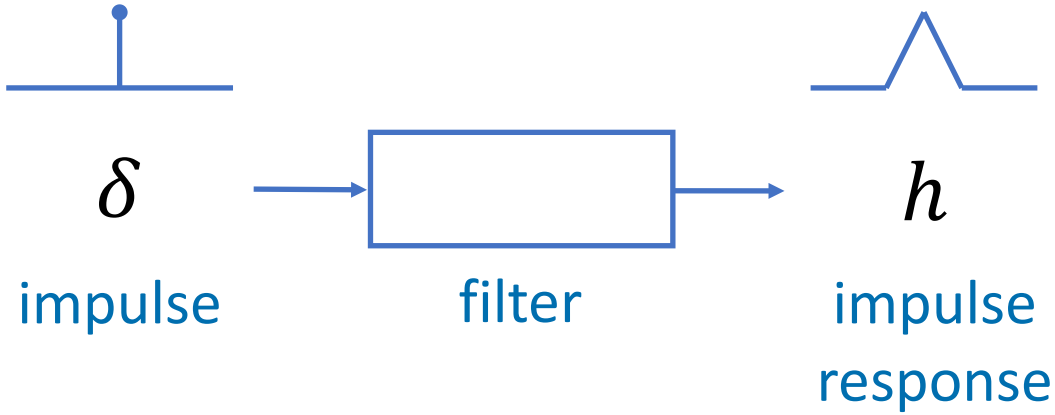
Filtering

- You can apply filtering to 1D, 2D, 3D or even higher-dimensional signals.
    - Audio, speech
    - Natural images
    - Medical images
    - Radar images
    - Satellite images
    - …
- You can smooth, sharpen, denoise images.
- You can add special effects to music.

A lot of filtering in musical performance

Impulse response

- To mathematically describe a filter, we introduce the concept of impulse response.
- The impulse response is the output of a filter when the input is an impulse signal.



Impulse response

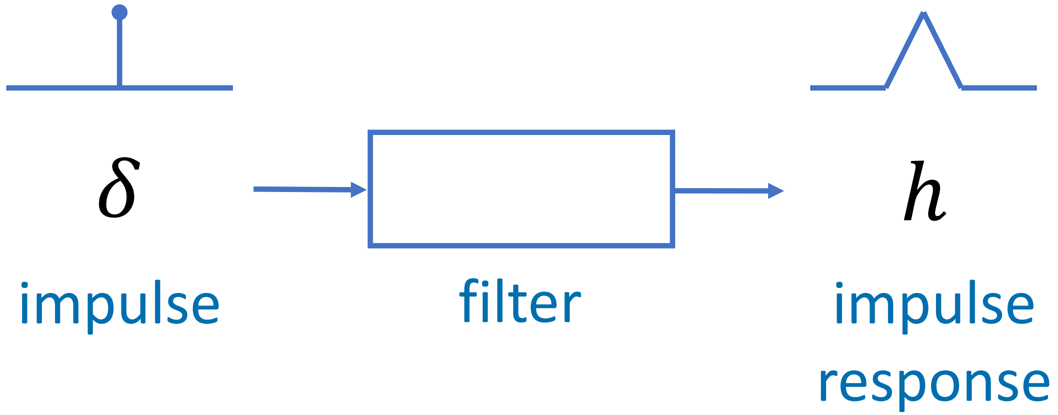
- For continuous signal, an impulse is a Dirac delta function $\delta(x)$,

$$\delta(x) = \begin{cases} +\infty, & if\ x = 0 \\ 0, & otherwise \end{cases}$$

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

- For discrete signal, an impulse is a Kronecker delta function $\delta[i]$,

$$\delta[i] = \begin{cases} 1, & if\ i = 0 \\ 0, & otherwise \end{cases}$$



Impulse response

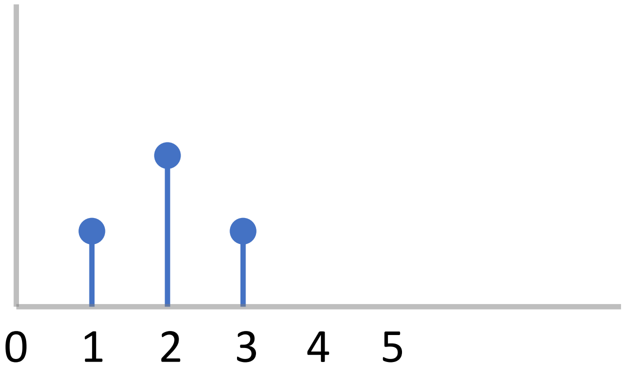
- The impulse response h completely characterises a *linear time-invariant* filter. As along as we know h, we can calculate the output signal y, given any input x.
- In signal processing, we often denote a filter by its impulse response function h.

$$x \longrightarrow \boxed{h} \longrightarrow y$$

input filter output

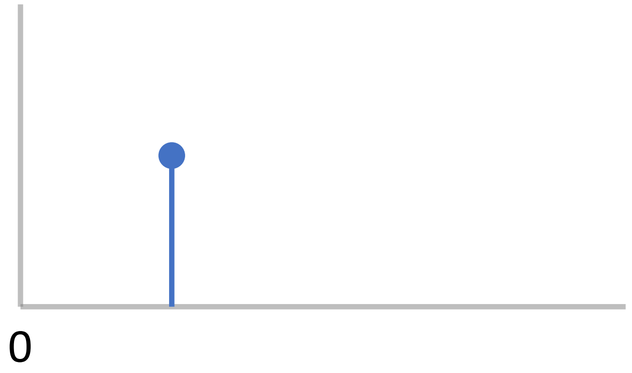
Impulse response

Input $f[n] = \delta[n]$

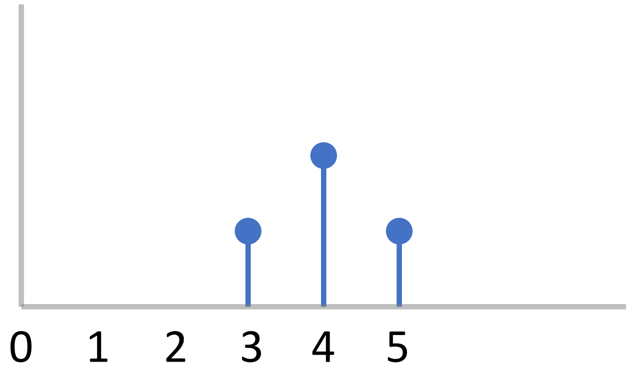


Output $g[n] = h[n]$

Time-invariant system

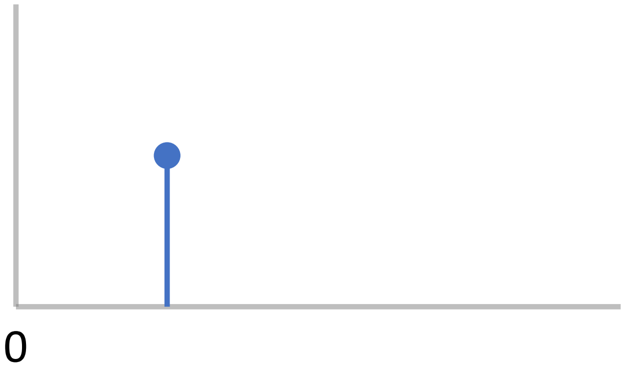


Input $f[n] = \delta[n-2]$

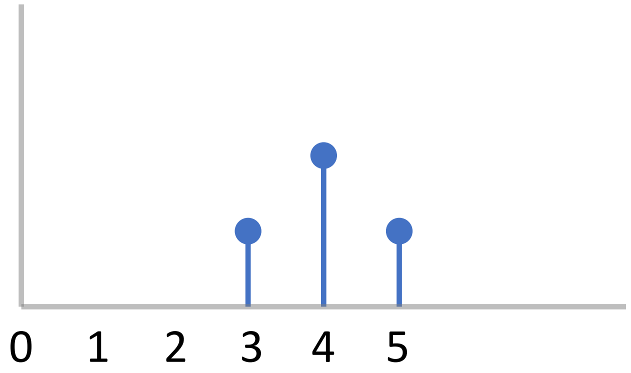


Output $g[n] = h[n-2]$

Time-invariant system



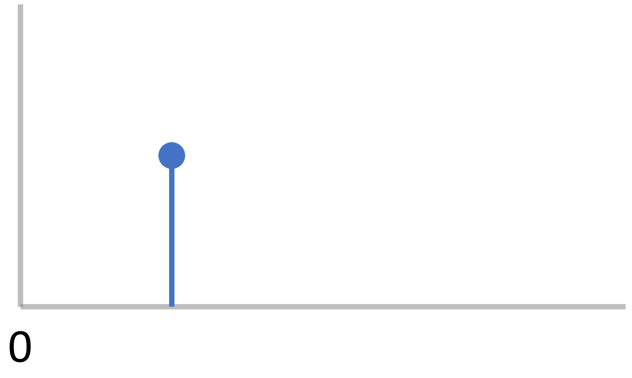
Input $f[n] = \delta[n-2]$



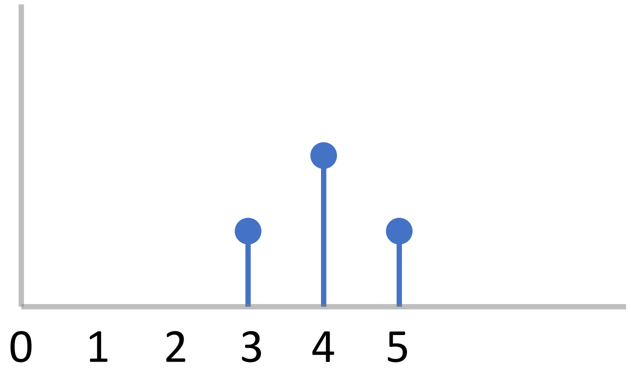
Output $g[n] = h[n-2]$

- If a filter is a time-invariant system, when we shift the input by time step k, the output will also shift by k.
- For example,
  - $g[n] = 10 \cdot f[n]$ is time-invariant, which amplifies the input by a constant.
  - $g[n] = n \cdot f[n]$ is not time-invariant, whose output depends on time step n.

Time-invariant system



Input $f[n] = \delta[n-2]$

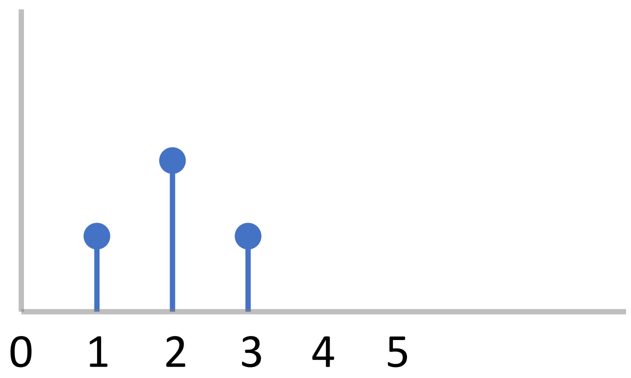


Output $g[n] = h[n-2]$

- If a filter is a time-invariant system, when we shift the input by time step k, the output will also shift by k.
- Because when we shift the input f by k, the output
  - $g[n] = 10 \cdot f[n-k]$ is a shift of $10 \cdot f[n]$.
  - $g[n] = n \cdot f[n-k]$ is not a simple shift of $n \cdot f[n]$.

Impulse response

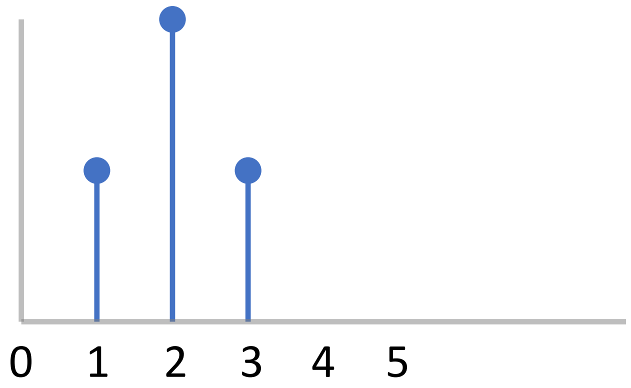
Input $f[n] = \delta[n]$



Output $g[n] = h[n]$

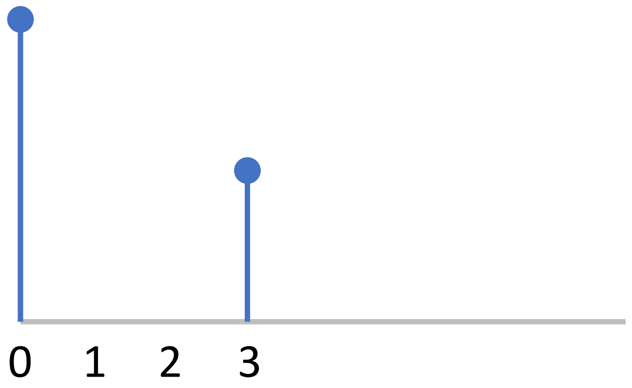
Linear system

Input $f[n] = 2 \cdot \delta[n]$

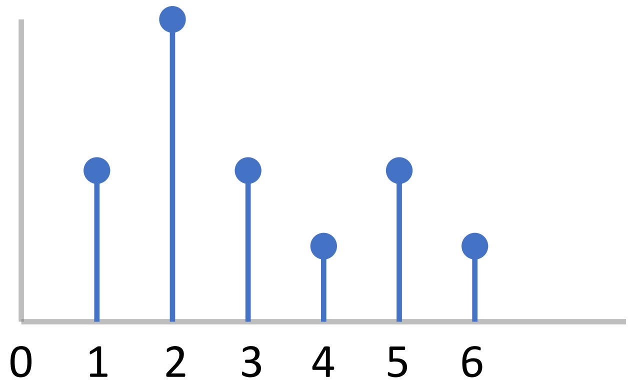


Output $g[n] = 2 \cdot h[n]$

Linear system

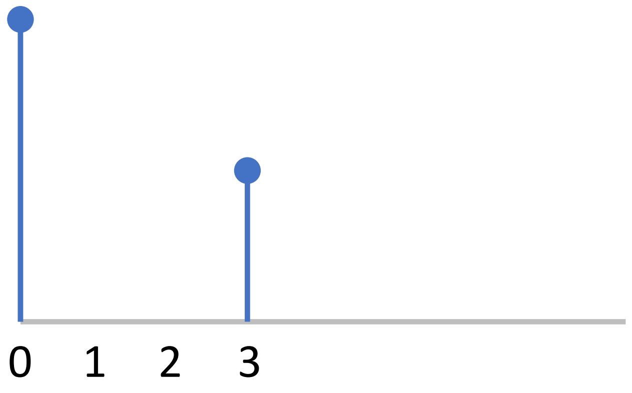


Input $f[n] = 2 \cdot \delta[n] + \delta[n-3]$

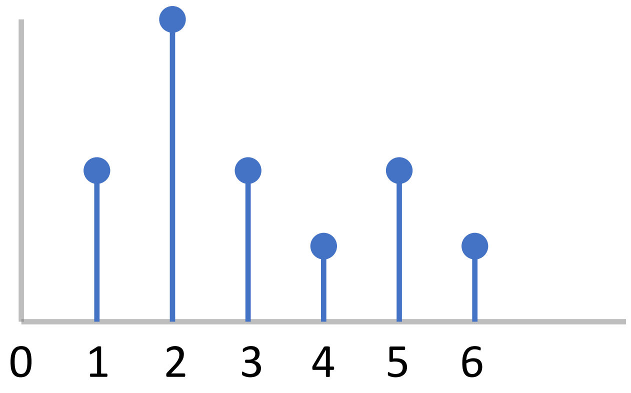


Output $g[n] = 2 \cdot h[n] + h[n-3]$

Linear system



Input $f[n] = 2 \cdot \delta[n] + \delta[n-3]$



Output $g[n] = 2 \cdot h[n] + h[n-3]$

- If a filter is a linear system, when we combine two input signals linearly, their outputs will also be combined linearly.
- For example, if we know input $f_1[n]$ leads to output $g_1[n]$, and input $f_2[n]$ leads to output $g_2[n]$, we will have

$$output(\alpha f_1[n] + \beta f_2[n]) = \alpha g_1[n] + \beta g_2[n]$$

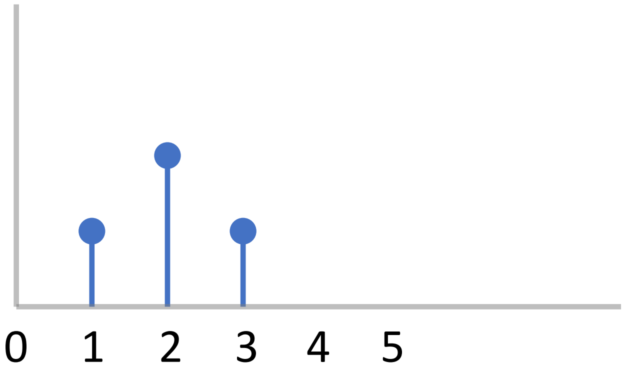
Linear time-invariant system

- Theories and methods are well developed for linear time-invariant systems.
- In our last lecture, the moving average filter, Gaussian filter and many other filters are linear time-invariant.
- For a linear time-invariant system, the impulse response h completely characterises how this system works. The output g can be described as the convolution between input f and impulse response h.

$$g[n] = f[n] \underbrace{*}_{\text{convolution}} h[n]$$

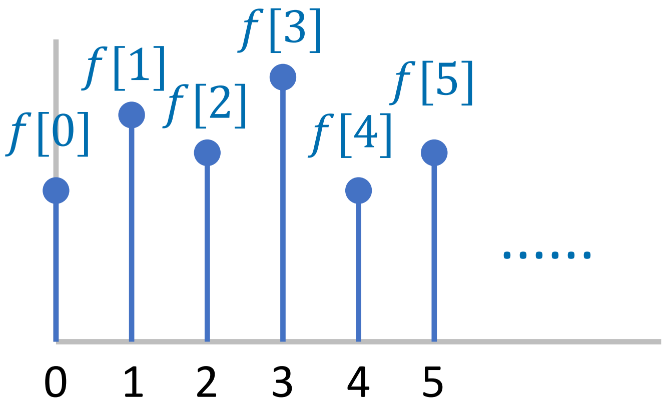
Impulse response

Input $f[n] = \delta[n]$

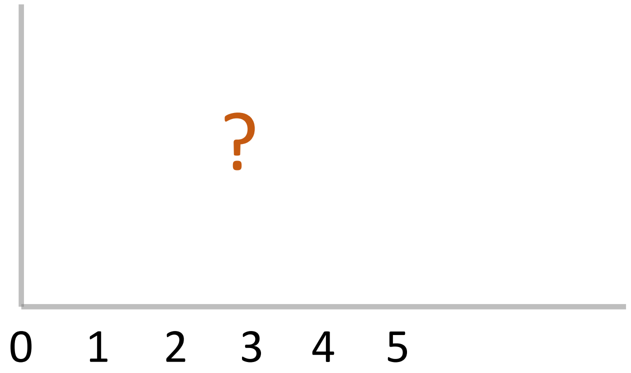


Output $g[n] = h[n]$

Linear time-invariant system

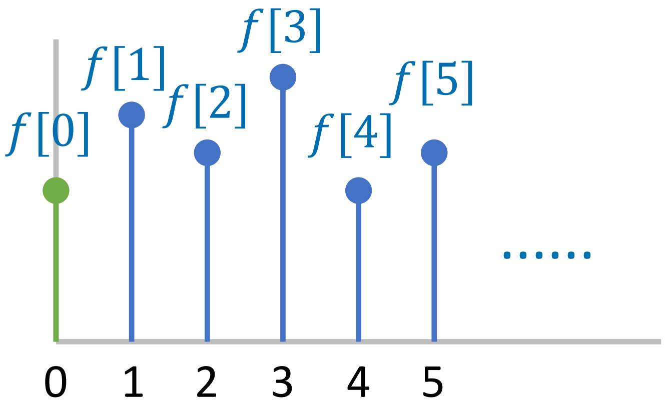


Input $f[n] = f[0]\delta[n] + f[1]\delta[n-1]$
$+f[2]\delta[n-2] + f[3]\delta[n-3] + \cdots$

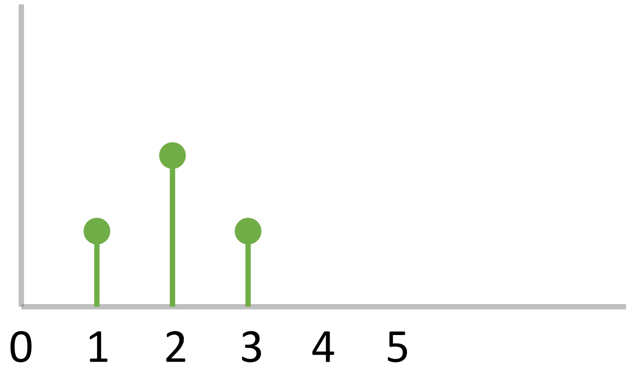


Output $g[n]$

Linear time-invariant system

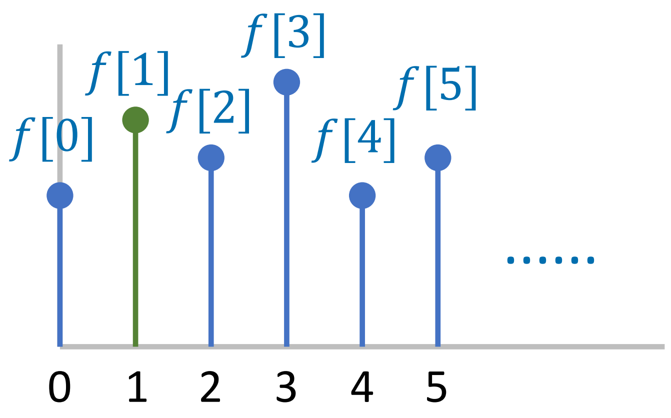


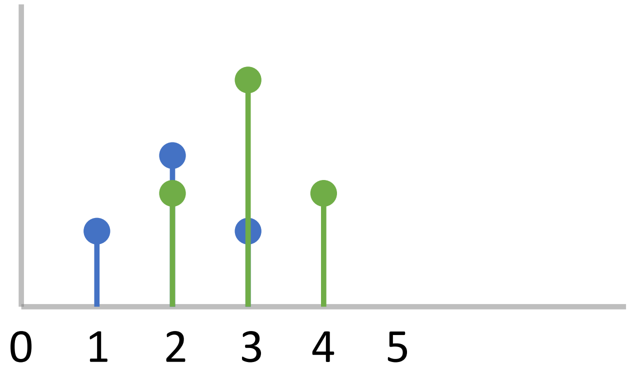
Input $f[n] = f[0]\delta[n] + f[1]\delta[n-1]$
$+f[2]\delta[n-2] + f[3]\delta[n-3] + \cdots$



Output $g[n] = f[0]h[n] + \cdots$

Linear time-invariant system

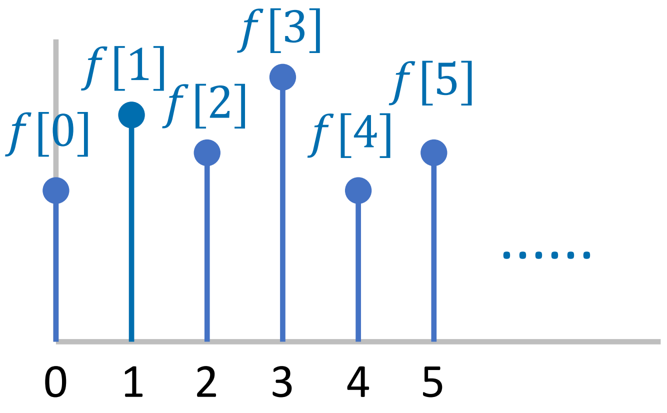


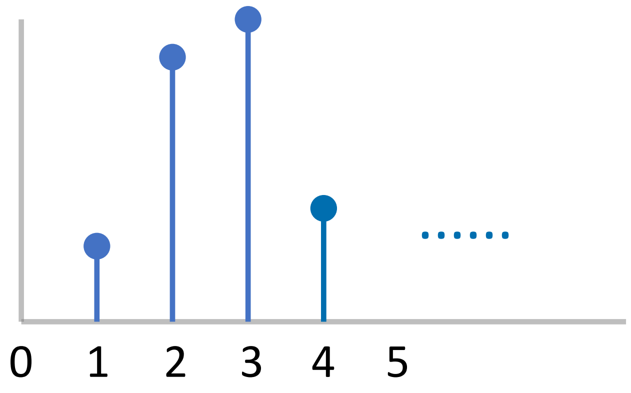


Input $f[n] = f[0]\delta[n] + f[1]\delta[n-1]$
$+f[2]\delta[n-2] + f[3]\delta[n-3] + \cdots$

Output $g[n] = f[0]h[n] + f[1]h[n-1]$
$+\cdots$

Linear time-invariant system





Input $f[n] = f[0]\delta[n] + f[1]\delta[n-1]$
$+f[2]\delta[n-2] + f[3]\delta[n-3] + \cdots$

Output $g[n] = f[0]h[n] + f[1]h[n-1]$
$+f[2]h[n-2] + f[3]h[n-3] + \cdots$

This mathematical operation is defined as convolution.

Convolution

- Convolution of a signal f and a filter with impulse response h is defined as,

$$g[n] = f[n] * h[n] = \sum_{m=-\infty}^{\infty} f[m]\underbrace{h[n-m]}_{\text{impulse response, convolution kernel}}$$

Commutativity

- We notice that

$$\sum_{m=-\infty}^{\infty} f[m]h[n-m] = \sum_{m=-\infty}^{\infty} f[n-m]h[m]$$

- This means that the convolution of f and h is equivalent to the convolution of h and f, known as commutativity,

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

Associativity

- Convolution satisfies the associativity,

$$f * (g * h) = (f * g) * h$$

- You can check the associativity by expanding the left equation.

$$f * (g * h) = \sum_{m=-\infty}^{\infty} f[m](g * h)[n - m]$$
$$= \sum_{m=-\infty}^{\infty} f[m] \sum_{k=-\infty}^{\infty} g[k]h[n - m - k] = \cdots$$

- Similarly expand the right equation $(f * g) * h$, using a scratch paper and using change of variables technique, you will find the associativity holds.

Properties of convolution

- Commutativity

$$f * h = h * f$$

- Associativity

$$f * (g * h) = (f * g) * h$$

- Distributivity

$$f * (g + h) = (f * g) + (f * h)$$

- Differentiation

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

Convolution

- In signal processing, convolution is often implemented this way.

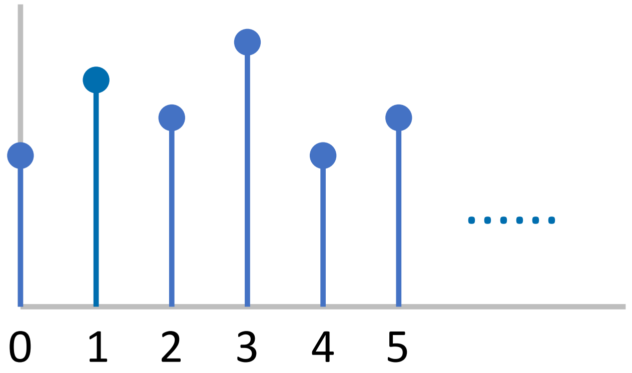
$$g[n] = f[n] * h[n] = \sum_{m=-\infty}^{\infty} f[n-m]h[m] = \sum_{m=-\infty}^{\infty} f[n+m]h[-m]$$

step 1: flip the kernel

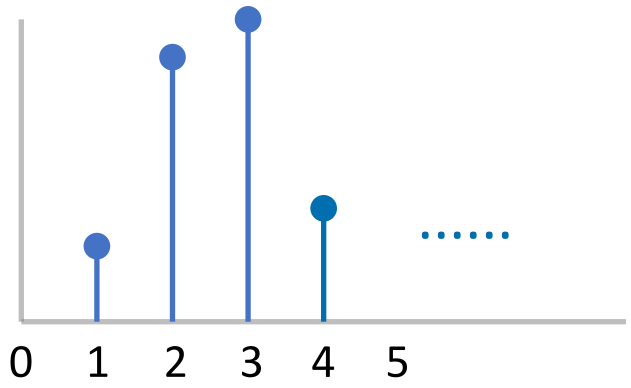
step 2: multiply the signal
with the flipped kernel

step 3: sum over the support
of the kernel

How to describe filtering mathematically?



Input $f[n]$



Output $g[n] = f[n] * h[n]$

$$= \sum_m f[n-m]h[m]$$

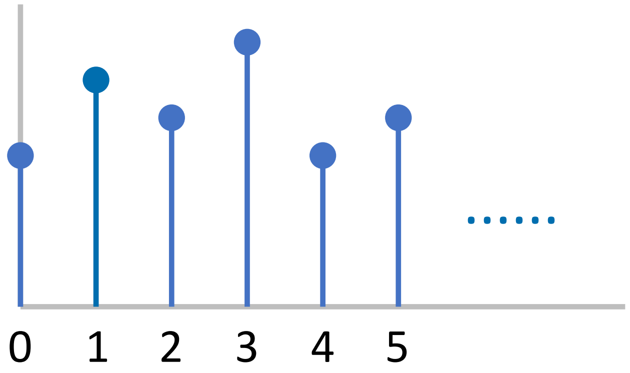
$$= f[n-1]h[1] + f[n]h[0] + f[n+1]h[-1] + \cdots$$

1/3	1/3	1/3

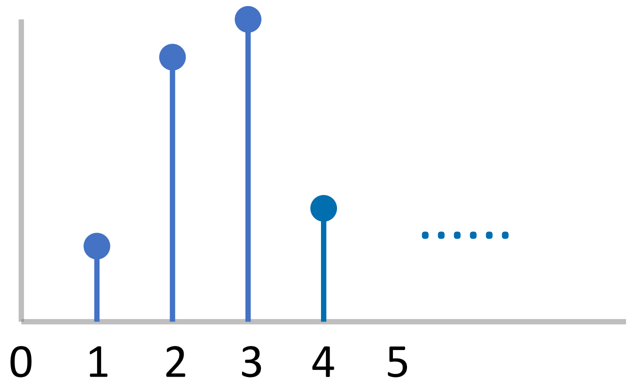
189	149	108	111	113	120	126	125

		126					

How to describe filtering mathematically?



Input $f[n]$



Output $g[n] = f[n] * h[n]$

$$= \sum_m f[n-m]h[m]$$

$$= f[n-1]h[1] + f[n]h[0] + f[n+1]h[-1] + \cdots$$

$h[1]$	$h[0]$	$h[-1]$

	$f[n-1]$	$f[n]$	$f[n+1]$				

		$g[n]$					

Filtering can be described as convolution $g[n] = f[n] * h[n]$.

In this example, the convolution kernel is $h[n]$ and $h[-1] = \frac{1}{3}$, $h[0] = \frac{1}{3}$, $h[1] = \frac{1}{3}$.

Filtering as convolution

- You can define $h[n] = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ and perform moving average filtering using the convolution operation

$$g[n] = f[n] * h[n]$$

- Convolution is implemented in many programming languages, e.g. calling scipy.signal.convolve(*f*, *h*) in Python.

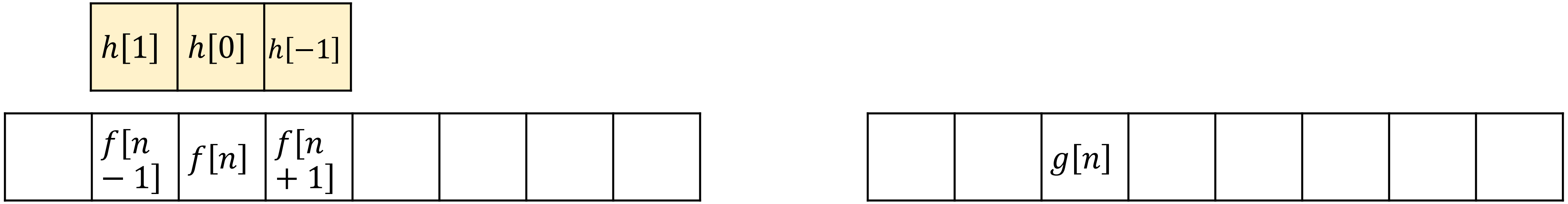


Image filtering

- After explaining the 1D case of filtering and convolution, now we show the 2D case, i.e. 2D image filtering.

Image filtering

Kernel h

1/9	1/9	1/9	111	110	123	130	130
1/9	1/9	1/9	111	113	120	126	125
1/9	1/9	1/9	108	113	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

Input $f[m,n]$

	147						

Output $g[m,n]$

2D convolution

- Convolution of 2D signal f with kernel h is defined as,

$$g[m,n] = f[m,n] * h[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j]h[m-i,n-j]$$

- It can also be written as,

$$g[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[m-i,n-j]h[i,j]$$

replace $m-i, n-j$ by i, j

$$= f[m-1,n]h[1,0] + f[m,n]h[0,0] + f[m+1,n]h[-1,0] + \cdots$$

2D convolution

- In signal processing, 2D convolution is often implemented this way.

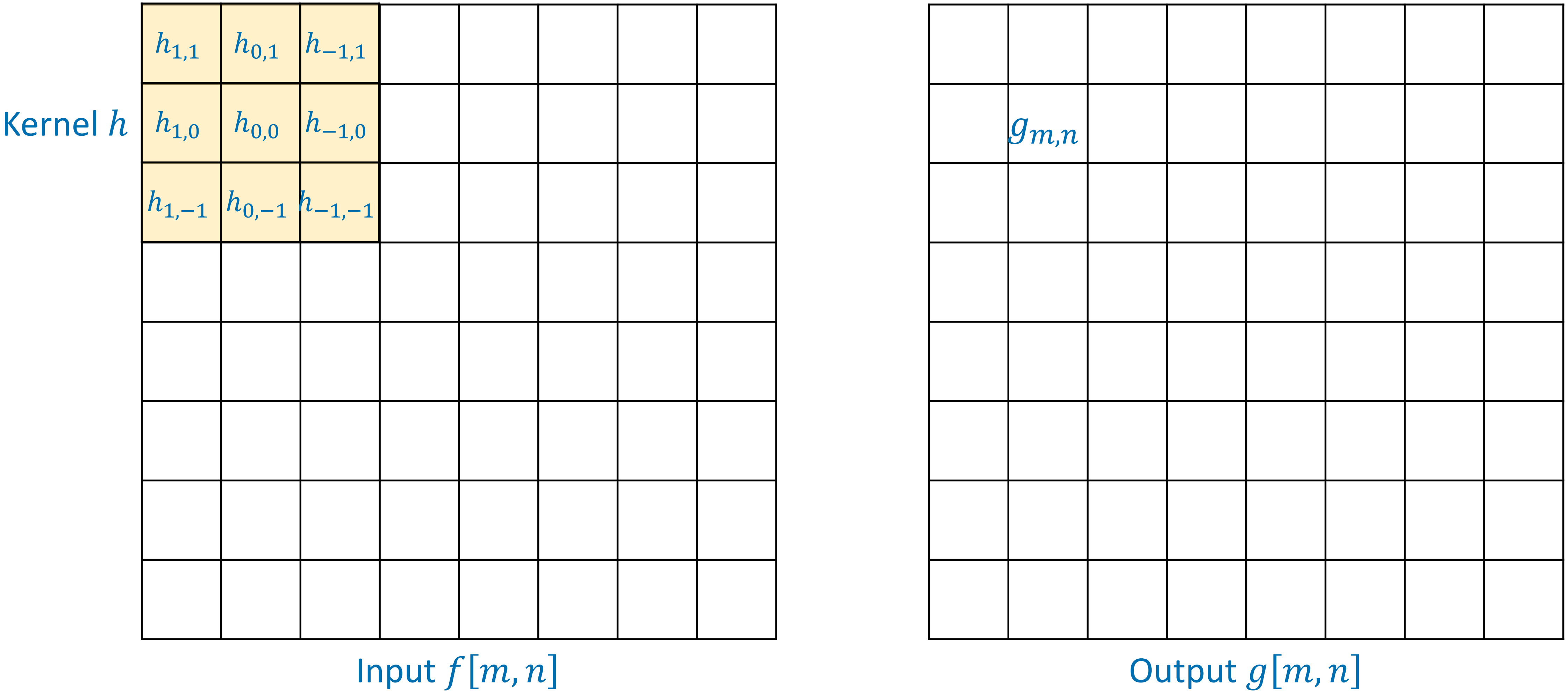
$$g[m,n] = f[m,n] * h[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[m+i, n+j] \cdot h[-i,-j]$$

step 1: flip the kernel both horizontally and vertically

step 2: multiply the image patch centred at pixel (m,n) with the flipped kernel

step 3: sum over the support of the kernel

2D convolution



2D convolution

- When you define the kernel $h[m, n]$ to be $\begin{array}{|c|c|c|}\hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline \end{array}$,

  moving average can be performed by using the convolution operation

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

Properties of convolution

- Commutativity

$$f * h = h * f$$

- Associativity

$$f * (g * h) = (f * g) * h$$

- Distributivity

$$f * (g + h) = (f * g) + (f * h)$$

- Differentiation

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

Associativity

- Associativity property of convolution
$$f * (g * h) = (f * g) * h$$
- If a big filter can be separated as the convolution of two small filters, such as $g * h$, then we can first convolve f with g, then with h.
$$f * filter_{big} = f * (g * h) = (f * g) * h$$

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

$filter_{big}$

$=$

0	0	0
1/3	1/3	1/3
0	0	0

g

$*$

0	1/3	0
0	1/3	0
0	1/3	0

h

We used this property for separable filtering.

Associativity

- Associativity property of convolution

$$f * (g * h) = (f * g) * h$$

- If a big filter can be separated as the convolution of two small filters, such as $g * h$, then we can first convolve f with g, then with h.

$$f * filter_{big} = f * (g * h) = (f * g) * h$$

$$\underset{filter_{big}}{\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}} = \underset{g}{\begin{bmatrix} & & \\ 1/3 & 1/3 & 1/3 \\ & & \end{bmatrix}} * \underset{h}{\begin{bmatrix} & 1/3 & \\ & 1/3 & \\ & 1/3 & \end{bmatrix}}$$

We used this property for separable filtering.

Image filtering and convolution

- In this lecture, we provide the mathematical foundation for image filtering.
- For a linear time-invariant system, filtering can be described by a mathematical operation, called convolution.
- Why do we introduce convolution?
  - So you can relate the knowledge here to signal processing.
  - So you know the existing convolution functions in Python or Matlab libraries can be called for image filtering.
  - You may need this to understand other concepts in image processing or your future study.

References

- Section 3.2: Linear filtering; Section 3.3.1: Non-linear filtering. Richard Szeliski, Computer Vision: Algorithms and Applications (http://szeliski.org/Book).