Filtering Theory

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- This lecture reviews filtering theory.
  - Linearity and spatial-invariance (LSI)
  - Impulse response
  - Sifting integral
  - Convolution

Definitions

- We use two criteria for filters: linearity and spatial-invariance

$$\begin{tikzcd} f(x) \arrow[r] & \boxed{\text{Filter}} \arrow[r] & g(x) \end{tikzcd} \qquad \begin{tikzcd} f(x) \arrow[r] & g(x) \end{tikzcd}$$

Linear :

$\alpha f(x) \rightarrow \alpha g(x)$ output is proportional to input

$f_1(x) + f_2(x) \rightarrow g_1(x) + g_2(x)$ superposition

or simply,

$\alpha f_1(x) + \beta f_2(x) \rightarrow \alpha g_1(x) + \beta g_2(x)$

Space - invariant(shift - invariant) :

$f(x-\lambda) \rightarrow g(x-\lambda)$

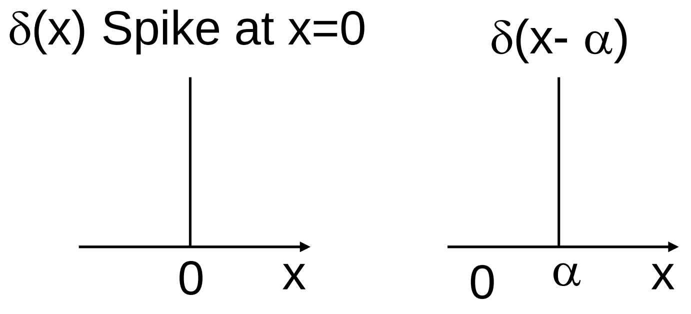
LSI Filter

- Physically realizable filters (lenses) are rarely LSI filters.
  - Optical systems impose a limit in their maximum response.
  - Power can't be negative, imposing a limit on the minimum response.
  - Lens aberrations and finite image area prevent LSI property.
- Nevertheless, close enough to approximate as LSI.

Impulse Response

$$\delta(x) \longrightarrow \boxed{\text{Filter}} \longrightarrow h(x)$$

$$\delta(x) = \begin{cases} \lim\limits_{\varepsilon \to 0} \int_{-\varepsilon}^{\varepsilon} f(x')dx' = 1, & x = 0 \\ 0, & x \neq 0 \end{cases}$$



δ(x) is the impulse function, or Dirac delta function which is a continuous function.

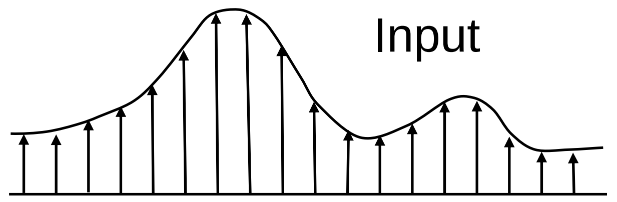
$$f(x_0) = \int_{-\infty}^{\infty} f(\lambda)\delta(x_0 - \lambda)d\lambda \qquad \delta(x_0 - \lambda) \text{ samples } f(x) \text{ at } x_0$$

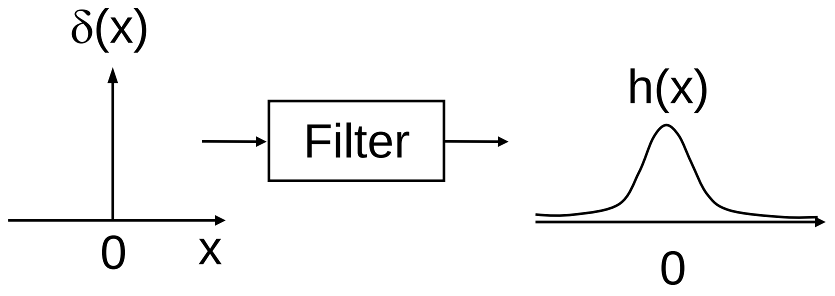
$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & x \neq 0 \end{cases}$$ **Kronecker delta function (discrete case: integer values of x)**

$$f(x) = \int_{-\infty}^{\infty} f(\lambda)\delta(x - \lambda)d\lambda$$ **Sifting integral**

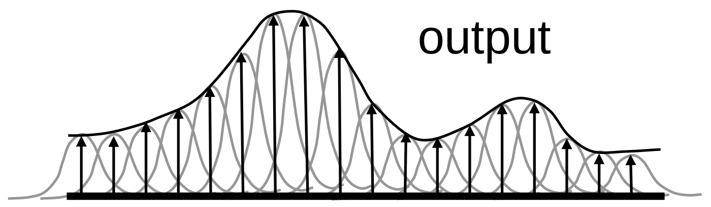
Convolution

- Any input signal can be represented by an infinite sum of shifted and scaled impulses:





Convolution:



Blurred version of input

$$\text{continuous}:\ g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\lambda)h(x-\lambda)d\lambda$$

Discrete Case

- Output function is a scaled shifted version of impulse response.

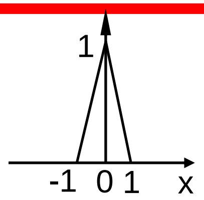
  ⊗, *: convolution operator

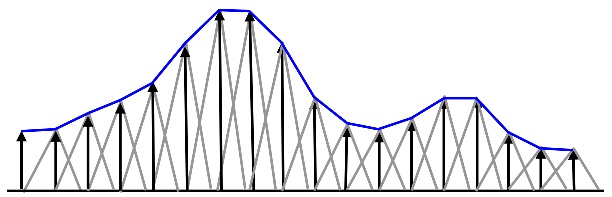
  h(x): convolution kernel, filter kernel
- If h(x) = δ(x) then we have an ideal filter: output = input.
- Usually h(x) extends over several neighbors.
- Discrete convolution:

$$g(x) = f(x) \otimes h(x) = \sum_{-\infty}^{\infty} f(\lambda) h(x-\lambda)$$

Example: Triangle Filter

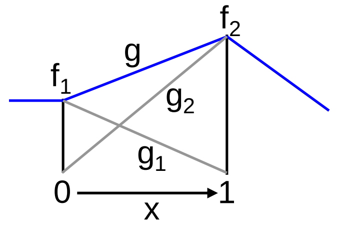
$$h(x) = \begin{cases} 1-|x| & 0 \le |x| < 1 \\ 0 & 1 \le |x| \end{cases}$$





Input samples (impulses)

Impulse responses (additive)



$$g_1 = f_1(1-x) \qquad g_2 = f_2 x \qquad 0 \le x \le 1$$

$$g = g_1 + g_2 = (f_2 - f_1)x + f_1$$

Addition of two adjacent impulse responses

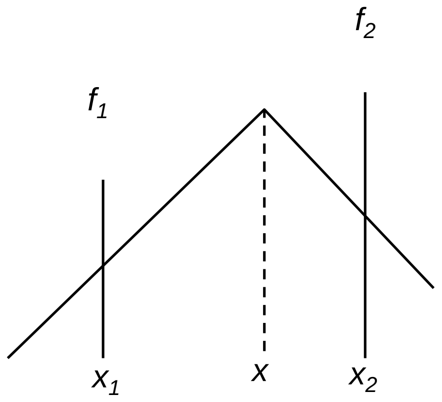
Straight line: y=mx+b

- Linearity rule: scale h(x) according to f(x) and add g_1, g_2.
- Obtain f(x) for <u>any</u> x by sampling the reconstructed g(x).

Convolution Summation

- g(x) is a continuous convolution summation to compute at particular values at x.

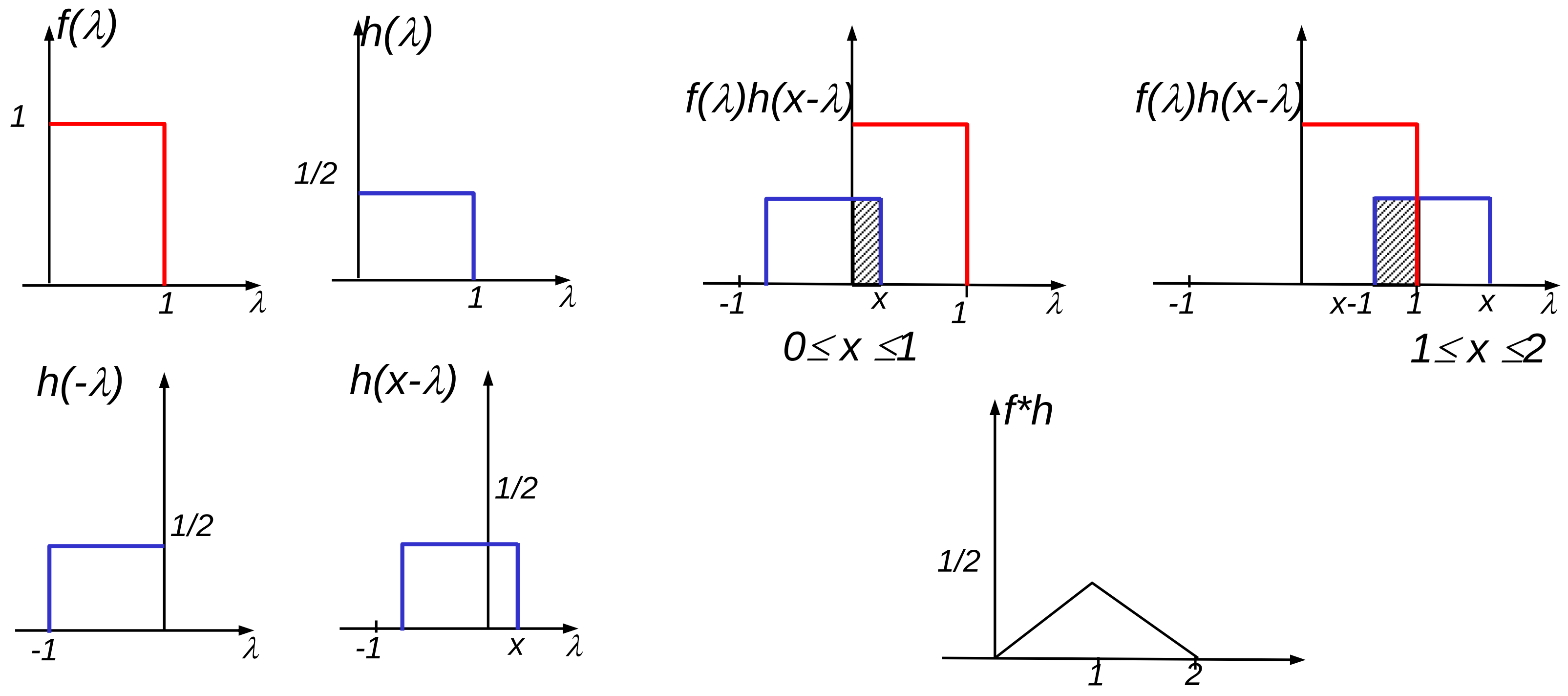
$$g(x) = f(x) \otimes h(x) = \sum_{-\infty}^{\infty} f(\lambda) h(x-\lambda)$$



$g(x) = f_1 h(x-x_1) + f_2 h(x-x_2)$

If $x_1=0$ and $x_2=1$ then $g(x)=f_1(1-x)+f_2 x$

A Closer Look At the Convolution Integral



Mirrored Kernel

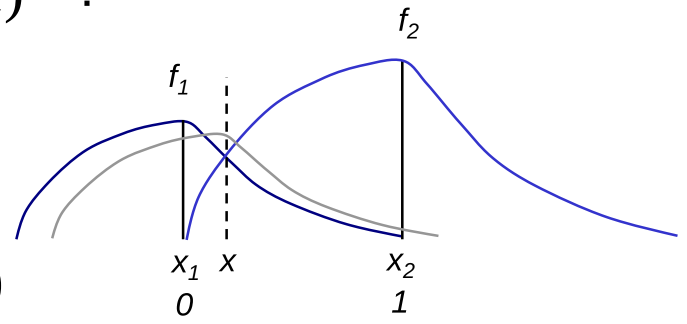
- Why fold over (mirror) kernel h(x)?
- Why not use $g(x) = \sum_{-\infty}^{\infty} f(\lambda)h(\lambda - x)$?

$$\text{By construction: } f_1 h(x - x_1) + f_2 h(x - x_2)$$

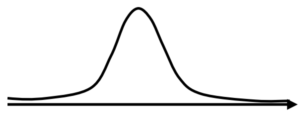
$$\sum f(\lambda)h(x - \lambda)$$

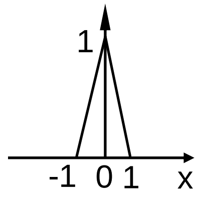
$$\text{By centering } h \text{ at } x: f_1 h(x_1 - x) + f_2 h(x_2 - x)$$

(which is wrong) Therefore flip h before centering at x.



- We typically use symmetric kernels: h(-x) = h(x)

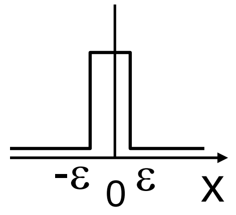




Impulse Function

- Impulse function (or Dirac delta function) is defined as

$$\delta(x) = \begin{cases} \lim\limits_{\varepsilon \to 0} \int_{-\varepsilon}^{\varepsilon} f(x')dx' = 1, & x = 0 \\ 0, & x \neq 0 \end{cases}$$



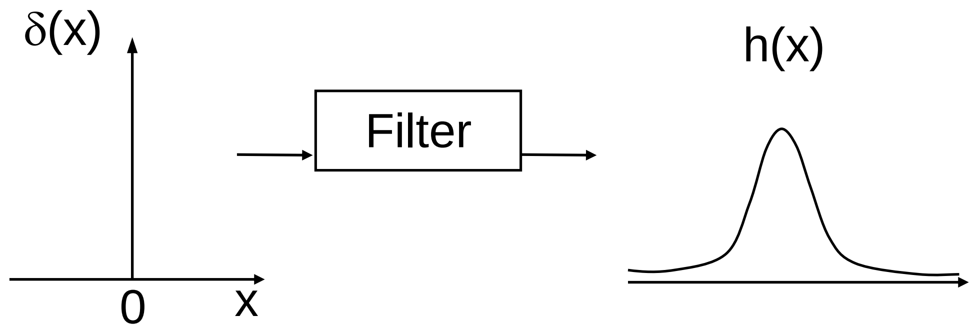
- It can be used to sample a continuous function *f(x)* at any point *x₀*, as follows:

$$f(x_0) = \int_{-\infty}^{\infty} f(\lambda)\delta(x_0 - \lambda)d\lambda = f(x_0)\delta(0) = f(x_0)$$

$$\delta(x_0 - \lambda) = 0 \text{ for } \lambda \neq x_0$$

Impulse Response

- When an impulse is applied to a filter, an altered impulse, (the *impulse response)* is generated at the output.



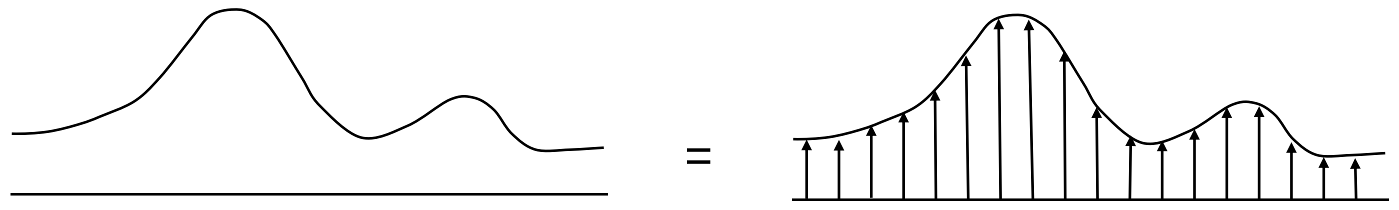
- The first direct outcome of LSI is that the filter can be uniquely characterized by its impulse response.

Sifting Integral

- Any continuous input signal can be represented in the limit by an infinite sum of shifted and scaled impulses.
- This is an outcome of the *sifting integral*:

$$f(x) = \int_{-\infty}^{\infty} f(\lambda)\delta(x-\lambda)d\lambda$$

which uses signal *f(x)* to scale the collection of impulses:



Convolution Integral (1)

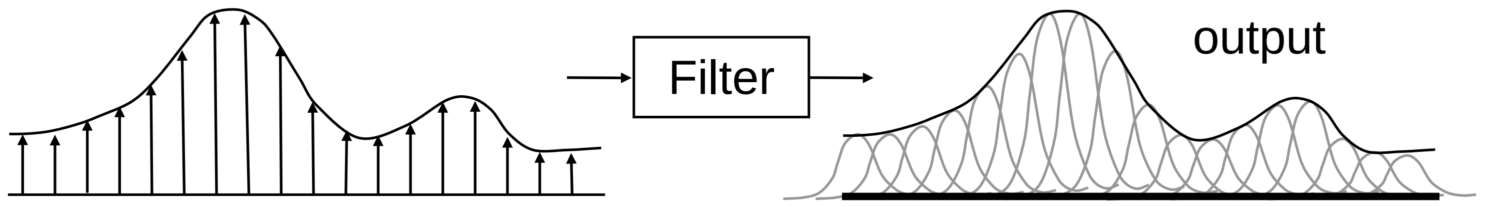
- The response *g(x)* of a digital filter to an arbitrary input signal *f(x)* is expressed in terms of the impulse response *h(x)* of the filter by means of *convolution integral*:

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\lambda) h(x-\lambda) d\lambda$$

  - where * denotes the convolution operation,
  - *h(x)* is used as the *convolution (filter) kernel*, and
  - λ is the dummy variable of integration.
- Kernel *h(x)* is treated as a sliding window that is shifted across the entire input signal.
- As *h(x)* makes its way across *f(x),* a sum of the pointwise products between the two functions is taken and assigned to output *g(x).*

Convolution Integral (2)

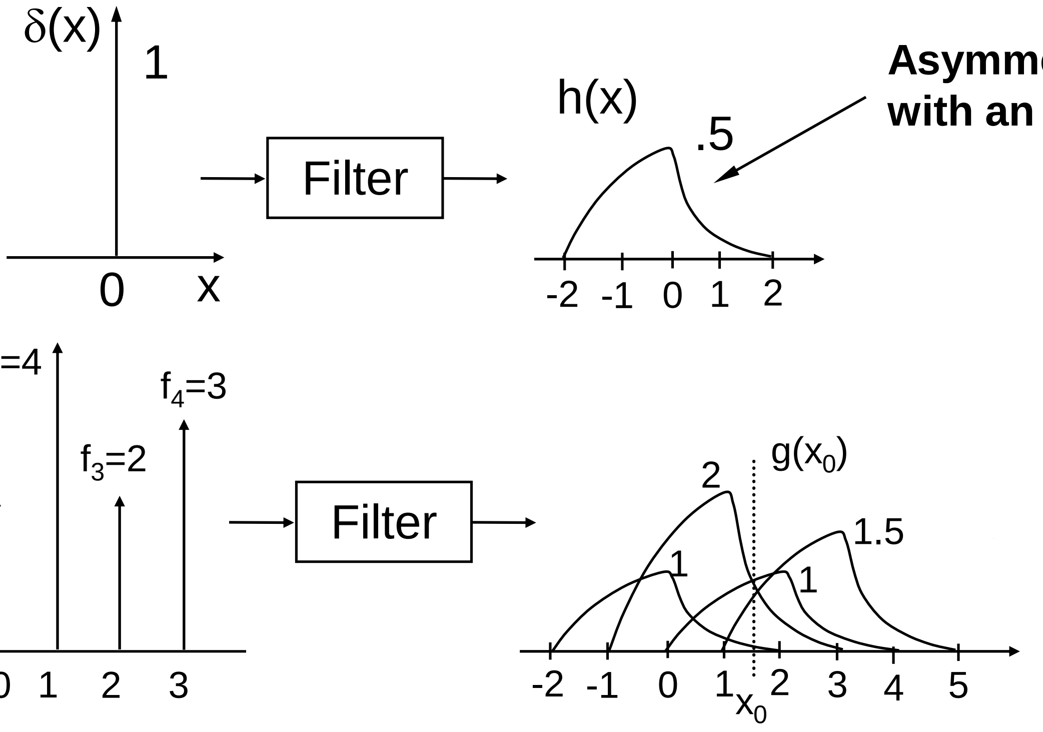
- This process, known as *convolution*, is of fundamental importance to linear filtering theory.
- It simply states that the output of an LSI filter will be a superposition of shifted and scaled impulse responses.
- This is used to explain how a continuous image is blurred by a camera as it passes through the lens.
  - In this context, *h(x)* is known as the point spread function (PSF), reflecting the limitation of the camera to accurately resolve a small point without somewhat blurring it.

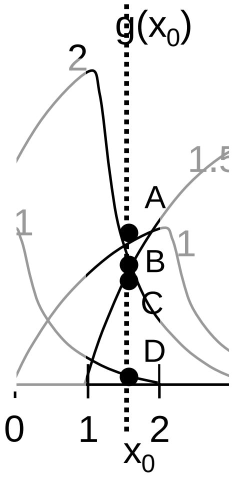


Evaluation

- We can arrive at *g(x)* in two ways:
  - Graphical construction of a series of shifted/scaled impulse responses
  - Computing the convolution summation at all points of interest
- Graphical construction more closely follows the physical process as an input signal passes through a filter.
- Convolution summation more closely follows the practical evaluation of *g(x)* at a finite number of desired points.
  - instead of adding scaled and shifted unit impulse responses (responses of unit impulses), we center the unit impulse response at the point of interest.

Graphical Construction

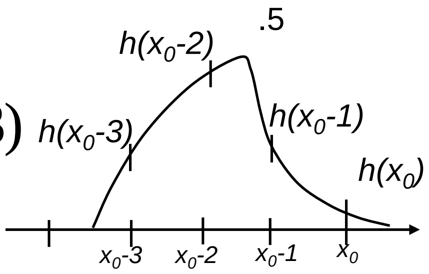




Blowup of interval between 1 and 2

$$g(x_0) = f_1 h(x_0) + f_2 h(x_0 - 1) + f_3 h(x_0 - 2) + f_4 h(x_0 - 3)$$

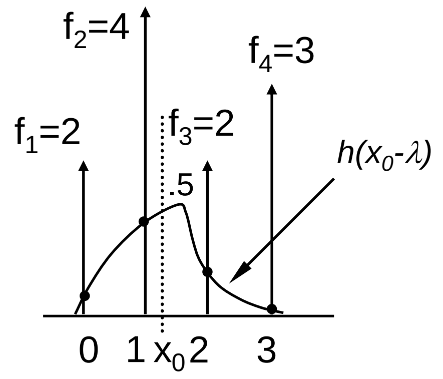
$$g(x_0) = 2h(x_0) + 4h(x_0 - 1) + 2h(x_0 - 2) + 3h(x_0 - 3)$$



As we scan f_i from left to right we multiply samples with values taken from h in right to left order.

Convolution Summation (1)

- Instead of adding scaled/shifted unit impulse responses, center unit impulse response at point of interest:

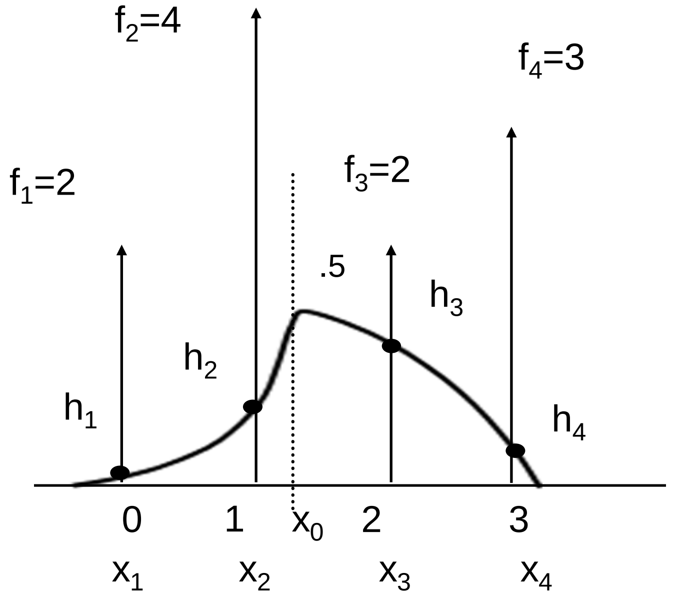


$$g(x_0) = \int_{-\infty}^{\infty} f(\lambda) h(x_0 - \lambda) d\lambda$$

- As λ increases, we scan *f()* from left to right. However, $h(x_0 - \lambda)$ is scanned from right to left.
- Reason: points to the left of x_0 had contributed to x_0 through the right side of *h* (see graphical construction).

Convolution Summation (2)

- More straightforward: implement convolution if both the input and the kernel were scanned in the same direction.
- This permits direct pointwise multiplication among them.
- Thus, flip kernel before centering it on output position.



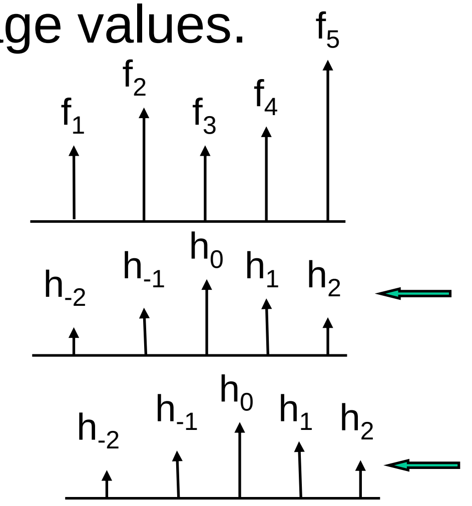
Convolution Summation (3)

- $\sum_{i=1}^{4} f_i h_i$ will yield the value for $g(x_0)$, where $h_i = h(x_i - x_0)$.
- Rationale: multiplying the unit impulse response *h* with f_i, *h* is being scaled. The distance between x_0 and x_i accounts for the effect of a shifted response function on the current output.
- For most impulse response functions, *h* will taper off with increasing distance from its center.
- If the kernel is symmetric ($h(x)=h(-x)$), it is not necessary to flip the kernel before centering.

Discrete Convolution (1)

$$g(x) = \sum_{-\infty}^{\infty} f(\lambda)h(x-\lambda)$$ for integer values of λ and arbitrary values of x

- The kernel is often a discrete set of weights, i.e., a *3x3* filter kernel.
- As long as kernel is shifted in pixel (integer) increments across the image, there is no alignment problem with underlying image.
- However, for noninteger pixel increments the kernel values may have no corresponding image values.



h is aligned with *f*
(*h* is centered at pixel f_3)

h is not aligned with *f*
It has been shifted by a ½-pixel increment

Discrete Convolution (2)

There are two possible solutions to this problem:

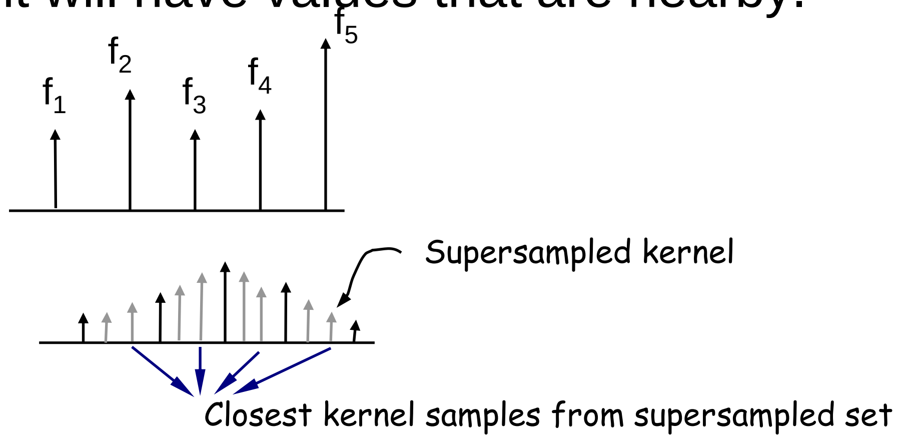
1. Represent h in analytic form for evaluation anywhere

   Ex: Let bell-shape PSF be $h(x)=2^{-4x^2}$. The appropriate weight to apply to f_i can be obtained by setting x to the difference between the center of the bell (impulse response) and the position of the f_i.

2. Supersample *h* so that a dense set of samples are used to represent *h.* The supersampled version will then be aligned with the image data, or at least it will have values that are nearby.

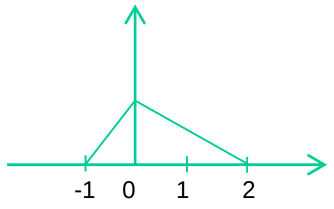
If the kernel is known to be slid across the image at fixed increments, then the kernel can be sampled at known positions.

Ex: a 1/3 increment requires the kernel to Be sampled three times per unit interval.

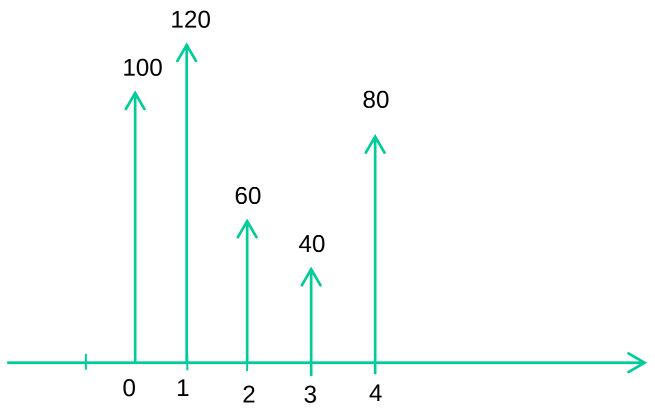


Example

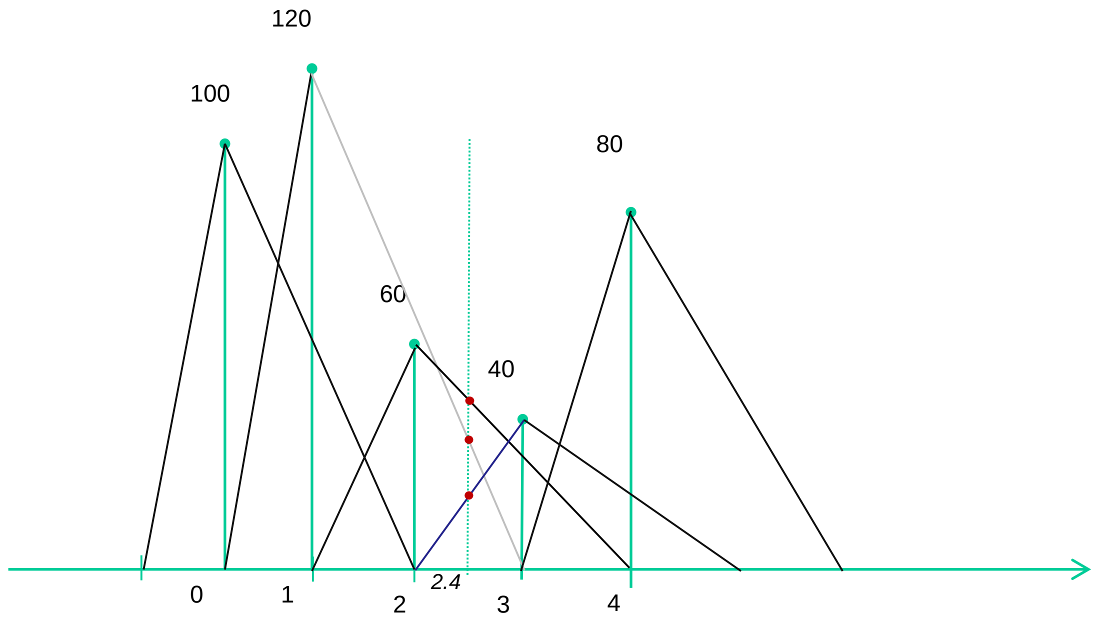
Kernel h(x) = $$\begin{cases} x+1 & -1 \le x < 0 \\ 1-\frac{x}{2} & 0 \le x \le 2 \\ 0 & otherwise \end{cases}$$



Input:



Graphical Construction Method

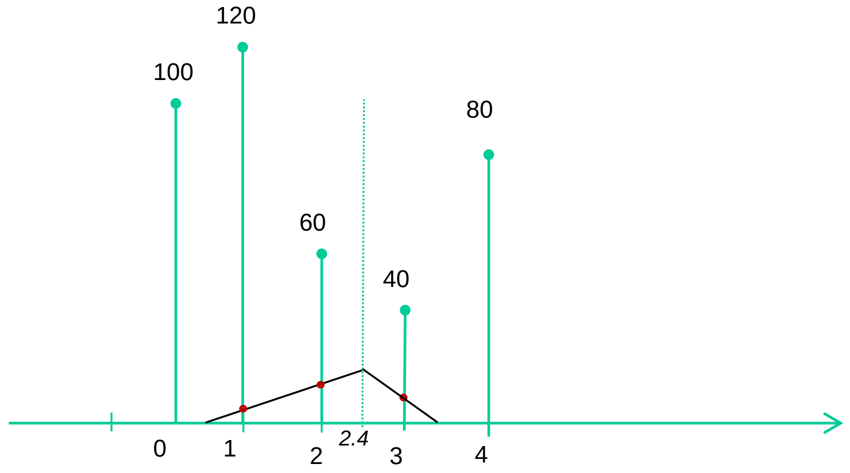


$$g(2.4) = f_1h(2.4-1) + f_2h(2.4-2) + f_3h(2.4-3)$$

$$g(2.4) = 120h(1.4) + 60h(0.4) + 40h(-0.6)$$

$$g(2.4) = 120(1-\frac{1.4}{2}) + 60(1-\frac{0.4}{2}) + 40((-0.6)+1)$$

Center Flipped Kernel at Position of Interest



$$g(2.4) = f_1 h(1-2.4) + f_2 h(2-2.4) + f_3 h(3-2.4)$$

$$g(2.4) = 120h(-1.4) + 60h(-0.4) + 40h(0.6)$$

$$g(2.4) = 120h(1.4) + 60h(0.4) + 40h(-0.6) \qquad (\textit{flipped})$$

$$g(2.4) = 120(1-\frac{1.4}{2}) + 60(1-\frac{0.4}{2}) + 40((-0.6)+1)$$

$$g(2.4) = 100$$