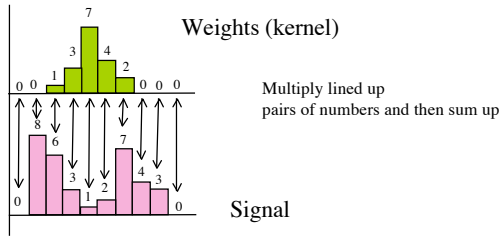
Syllabus Notes

- Filters is next. We will do §7.1, in less detail §7.1-7.4, in more detail §7.5 and §7.6, and mention some of §7.7.
- Then onto edge detection. We will touch on much of §8.

Linear Filters (§7)

- General process:
  - Form new image whose pixels are a **weighted sum** of original pixel values, using the same set of weights at each point.
- Much like a 2D version of the sensor response computation (sensitivities are like weights), but now compute a similar weighted sum for each point



Linear Filters (§7)

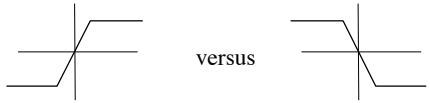
- Example: smoothing by averaging
  - form the average of pixels in a neighbourhood (weights are equal)
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood (weights follow a Gaussian function)
- Example: finding a derivative
  - negative weights on one side, positive ones on the other

Linear Filter Example

- Compute a new image which is an average of 3 by 3 blocks
- H (the kernel) $$H(i',j') = \begin{cases} \frac{1}{9} & i',j' \in [-1,1] \\ 0 & \text{otherwise} \end{cases}$$
- Result is $$R_{ij} = \sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} \frac{1}{9} F(i',j')$$

Linear Filters (§7)

- Properties
  - Output is a **linear** function of the input
  - Can represent as a matrix (but usually do not)
- Terminology
  - Array of weights is referred to as the kernel (H)
  - (Sometimes referred to as "mask" or "template")
- Be aware of two forms
  - Correlation (more natural, often what we visualize)
  - Convolution (more commonly referred to, has some useful mathematical properties)
  - Convolution is correlation by a flipped kernel (if kernel is symmetric, then no difference)



Convolution

- Denote by $\otimes$ (will see others symbols!).
- Represent weights as a second image, H (the kernel)
- Pretend that images are padded to infinity with zeros (sums don't need limits)
- Then, the definition of discrete 2D convolution is:

$$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{uv}$$

- Notice weird order of indices (includes the flips)

Properties of $R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{uv}$

- Linear
- Commutative
- A**ssociative** (Can save CPU time!)

$$(A \otimes B) \otimes C = A \otimes (B \otimes C)$$

- Output is a s**hift-invariant** function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Converse of above is true: If a system is linear and shift invariant, then it is a convolution.

Shift invariant linear systems (§7.2)

- Shift invariant
  - Shift in the input means we simply shift the output
  - Example: Optical system response to a point of light
    - Light moves from center to edge, so does its image
- Linear shift invariant
  - Can compute the output due to complex input, based on the response to a single point input
    - Discrete version---function is zero everywhere except f(x,y)=1 (call that *box*(x',y'))
    - Continuous version---delta function
- f(x,y) is a linear combination of shifted versions of *box*(x',y')

Shift invariant linear systems (§7.2)

$$f(i,j) = \sum \sum box(i-u, j-v) f(u,v)$$

Box shifted by (u,v). Note subtraction!

$$\text{Response}(box(i,j)) = h(i,j)$$

$$\text{Response}(box(i-u, j-u)) = h(i-u, j-u)$$

$$\text{Response}(\text{f(i, j)}) = R_{ij} = \sum \sum \; h(i-u, j-v) f(u,v)$$

(Convolution by h)

Response as sum of basis functions (§7.2)

- The response is linear combination of shifted versions of the kernel
- The weights are the values of the function being convolved
- The shifted versions of the kernels form a basis over which the result image is constructed
- Thinking of an image as a weighted sum over a basis is a generally useful idea—we will come back to this when we do Fourier transforms (only touched upon in 2008).