========
10/01/08
========
Convolution continued:

d[t] -> S -> h[t]

- for arbitrary signal x[1..n]
        - Sum(i..N) x[i]*h[t-i] == y[1..N] or y[t] <is unclear>

h[-t-i] dot C << C is the area around x that is the size of h ?

Properties of Convolution: x[t] # h[t] << impulse response of the system, kernel, filter; width of h = support of the kernel [# is drawn as a star]
        a) x[t] # d[t] = x[t] (identity) << all-pass filter
        b) x[t] # kd[t] = kx[t] (scaling) << when k>1 is amplifier, when k<1 is attenuator
        c) x[t] # d[t+s] = x[t+s] (shifting << delay system
        d) Commutative: a[t] # b[t] = b[t] # a[t] (x -> A -> B -> y is a cascaded system)
        e) Association: a[t] # (b[t] # c[t]) = (a[t] # b[t]) # c[t]
        f) Distributive: a[t] # (b[t] # c[t]) = (a[t] # b[t]) + (a[t] # c[t])

####How do we design a filter?
ex: Blurring. What do I need to do to delta to make it blurred?
                We just spread it out and lower the height to maintain the area
                In other words, do a blurring function! :p
        This is called a low-pass filter

###Fourier Transform
* Any complex signal can be represented as a combination of cosine waves -- produces a basis (ie, x-y-z vectors)
x[t] //t is time or spacial domain
x[t] = SUM(i) a*Cos(f,p) //a = amplitude, f = frequency, p = phase
                        Some other restrictions on t as well, will

discuss later
This takes us to the frequency domain (in terms of f,a,p; two plots instead of just one: a(f) and p(f) instead of just x(t))

For 2D signals:
x[u,u] = SUM(i) a*Cos(f,p,o)
Now have two 3D plots
instead of a = g(f,o)
a = g(k,l) where f = sqrt(k^2 +l^2), o = tan^-1 (l/k)
this lets things make sense--orientation is actually orientation, amplitude is actually distance, etc. Good for visualization

a(f) tells you how much "sharpness" the signal has
p(f) tends to point out edges in time domain, but are generally easier to look at in the time domain anyway

So why is this called a low-pass filter?
suppose d[t] -F-> A[f]=k (a flat line)
x[t]=k -F-> D[f] (a single spike)
//They swap between them!! Neato!!
Called "Duality"

a[t] -F-> A[f] # b[t] -F-> B[f] ==> A[f] * B[f] //a convolution in time is a multiplication in frequency

In other words if you multiply by delta, you're multiplying by the identify and get it back again. So it's an "all pass" filter

Expansion in one domain is compression in the other.

So as we expand in our blurring function (making it wider), our frequency is becoming more compressed, thus getting rid of the "higher" frequencies
(our bandwidth is reduced ?). Hence low-pass -- we let the lower frequencies through.

A sequence of images going from original to more and more blurred == "Gaussian Pyramid"

* High Pass Filter:

I - (I # h) //subtract the blurred image--the low frequencies
= I # d - I # h
= I # (d - h) //so convolute by the negative of the low pass, and literally go the other direction

* Band Pass Filter:
B1 = G1 - G2 //take the base and subtract off the frequencies we don't want
Produces a "Laplacian Pyramid" (pyramid of frequency bands)

###2D Separability
h[i,j] = a[i] * b[j] //if you can break up a 2D signal into a product of independent 1D signals,
then is called 'Separable Filter".
ex: for image
x[i,j] # h[i,j] = y[i,j] = x[i,j] # a[i] # b[j]