B. The concept of convolution

B.1 Convolution

```
<< FrontEndVision`FEV`;
Show[Import["retinapsf.gif"], ImageSize -> 150];
```

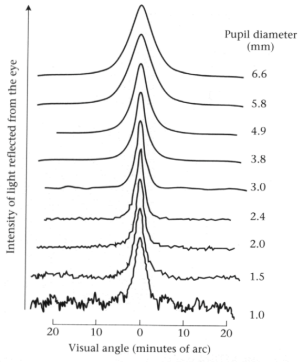

Figure B.1 Experimental measurements of light that has been reflected from a human eye looking at a fine line ('line spread function'). The reflected light has been blurred by double passage through the optics of the eye. The amount of unsharpness is a function of the pupil diameter: with wider pupil opening the line got more blurred. Source: Campbell and Gubish, 1996. Taken from [Wandell1995].

We take the example of the not-exactly-sharp projection of images on the human retina. Lens errors, cornea distortions, the retinal nerve tissue: they all contribute to the 'smearing effect'.

In fact, *every* point in the stimulus gives rise to the same blurring effect. Figure 1 shows some actual measurements at various pupil diameters. Suppose that the profile of the blur function is a Gaussian kernel. Then every point of the stimulus will become a Gaussian function when projected on the retina. We study the 1-D case, and call the blur function **g[x,σ]**. The blur function is also called the *filter* or the *kernel*, or the *operator* (sometime we encounter the name *stencil* or *template*).

$$g[x\_, \sigma\_] := \frac{1}{\sqrt{2\,\pi\,\sigma^2}}\,\text{Exp}\left[-\frac{x^2}{2\,\sigma^2}\right];$$

We define the input signal as **f[x]**, and the output signal as **g[x]**. We can think of the input signal as a series of points next to each other, and each of these input points gives rise to a blurred output point. We plot the input point function **fpnt[x]** (a very narrow function around the point **x=0**) at positions x=0 and x=2:

```
fpnt[x_] = If[-0.05 < x < 0.05, 1, 0];
Block[{$DisplayFunction = Identity},
p1 = Plot[fpnt[x], {x, -4, 4}]; p2 = Plot[fpnt[x - 2], {x, -4, 4}]];
Show[GraphicsArray[{p1, p2}], ImageSize -> 300];
```

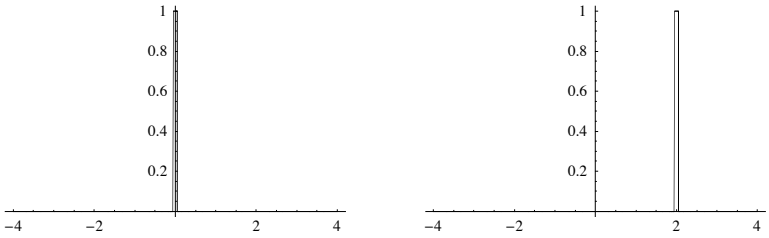

Figure B.2 The sampling function, at two arbitrary positions.

As an input signal we take an example function consisting of a sum of sine functions of different frequencies. We can approximate this function as a set of points very close to each other. The amplitude of each point is modulated with the input function, so we plot the product **f[α] fpnt[x-α]**.

```
f[x_] = Sin[x] + 0.5 Sin[5 x];
Block[{$DisplayFunction = Identity},
p1 = Plot[f[x], {x, -4, 4}];
  p2 = Show[Table[Plot[f[α] fpnt[x - α],
       {x, -4, 4}, PlotPoints -> 160], {α, -4, 4, .1}]]];
Show[GraphicsArray[{p1, p2}], ImageSize -> 350];
```

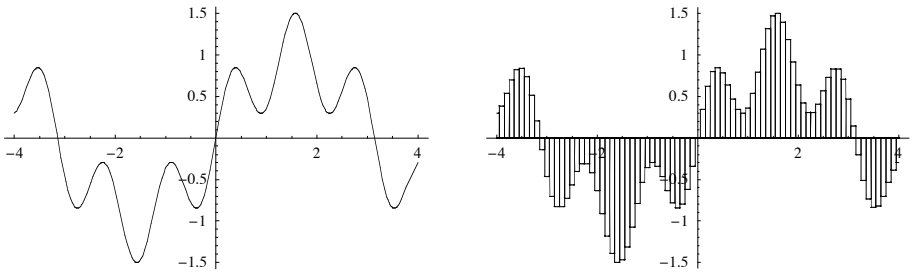

Figure B.3 Left: The input signal $f(x)$, right: the sampled representation.

Every point is blurred on its own, so we replace every pointfunction **fpnt[α]** by its blurred version **g[α]**. Every point is widened substantially, and the final step is adding all these responses together.

```
Block[{$DisplayFunction = Identity},
  p1 = Show[Table[Plot[f[α] gauss[x - α, 1],
       {x, -4, 4}, PlotPoints -> 160], {α, -4, 4, .1}]];
  h[x_] = Sum[f[α] gauss[x - α, 1], {α, -6, 6, .1}];
p2 = Plot[h[x], {x, -4, 4}]];
```

```
Show[GraphicsArray[{p1, p2}], ImageSize -> 350];
```

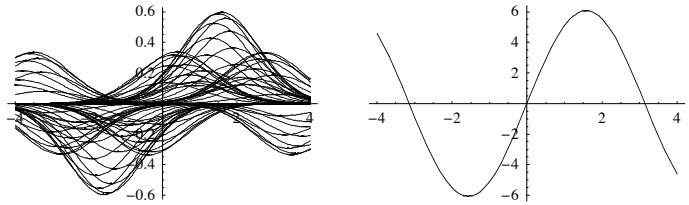

Figure B.4 Left: every sample gives rise to a kernel, at the respective position with the amplitude of the sample. Right: the summed response is the convolution.

In the limit of making the plot functions smaller and smaller we get a summation described by an integral: the convolution-integral. So we may write for the output **h[x]**:

$$h[x] = \int_{-\infty}^{\infty} f[\alpha]\, g[x-\alpha, \sigma]\, d\alpha$$

It describes exactly how the output **h[x]** looks like when an input signal **f[x]** is processed by a system **g[x]**. We say that the signal **f[x]** is filtered with a filter or kernel **g[x]**. We also call **g[x]** the pointspread-function of the system. Note that the convolution-integral integrates from -∞ to ∞, indicating that we integrate over the whole length of the signal. The parameter **α** is the so-called shift variable. (sometimes the convolution integral is called the shift-integral).

Actually, the sum above is an approximation. The exact output is given by the convolution-integral. Let us plot the integral, and draw your conclusions yourself. We have approximated the solution quite well:

```
h[x_] := ∫_-∞^∞ f[α] g[x - α, 1] dα;
p5 = Plot[Evaluate[h[x]], {x, -4, 4}, ImageSize -> 160];
```

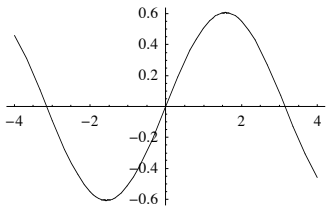

Figure B.5 The convolution as an exact integral of the analytical input function and kernel.

The output signal looks different from the input signal, and this is exactly what we wanted: we have filtered the high frequencies out of the signal. The frequency **Sin[5x]** is virtually eliminated, as the figure above is almost the low frequency **Sin[x]** function, as we put in the input. This low frequency filter is however 40% attenuated due to the filtering (check the amplitudes).

A short form for writing the convolution integral is the symbol ⊗ , the convolution operator. The function **h** is the convolution of the function **f** with **g**

```
h[x] = f[x] ⊗ g[x]
```

The convolution operator is a *linear* operator:

```
(h1[x] + h2[x]) ⊗ g[x] = h1[x] ⊗ g[x] + h2[x] ⊗ g[x];
(a f[x]) ⊗ g[x] = a (f[x] ⊗ g[x])
```

In 2-D we get:

$$h[x\_, y\_] := \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f[\alpha, \beta]\, g[x-\alpha, y-\beta]\, d\alpha\, d\beta$$

B.2 Convolution is a product in the Fourier domain

It is often not so easy to calculate the convolution integral. It is often much easier to calculate the integral of the convolution in the Fourier domain. This section explains how this works. We start from the convolution of the function $h(x)$ with the kernel $g(x)$:

$$f(x) = \int_{-\infty}^{\infty} h(y)\, g(x-y)\, dy$$

The Fourier transform $F(\omega)$ of $f(x)$ is then by definition

$$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(y)\, g(x-y)\, dy\, e^{-i\omega x}\, dx$$

We reorder the terms in the integral and make the substitution $x - y = \tau$, so we have $x = \tau + y$ and $e^{-i\omega x} = e^{-i\omega\tau}.\, e^{-i\omega y}$:

$$\int_{-\infty}^{\infty} h(y) \int_{-\infty}^{\infty} e^{-i\omega x}\, g(x-y)\, dx\, dy = \int_{-\infty}^{\infty} h(y) \int_{-\infty}^{\infty} e^{-i\omega(\tau+y)}\, g(\tau)\, d(\tau+y)\, dy$$

When we integrate to τ, we keep y constant, so $d(\tau+y)$ is equal to $d\tau$. So we get:

$$\int_{-\infty}^{\infty} h(y) \int_{-\infty}^{\infty} e^{-i\omega\tau}\, e^{-i\omega y}\, g(\tau)\, d\tau\, dy$$

Because $e^{-i\omega y}$ is constant in the integration to τ, we may bring it as a constant outside the integral:

$$\int_{-\infty}^{\infty} h(y)\, e^{-i\omega y}\, dy \int_{-\infty}^{\infty} e^{-i\omega\tau}\, g(\tau)\, d\tau = H(\omega)\,.\,G(\omega)$$

where $H(\omega)$ is the Fourier transform of the function $h(x)$ and $G(\omega)$ is the Fourier transform of the kernel $g(x)$. So the Fourier transform of a convolution is the product of the Fourier transform of the filter with the Fourier transform of the signal. To put it otherwise:

$$\begin{array}{ccccc} f(x) & = & h(x) & \otimes & g(x) \\ \updownarrow & & \updownarrow & & \updownarrow \\ F(\omega) & = & H(\omega)\,.\,G(\omega) & & \end{array}$$

We can calculate a convolution $h \otimes g$ by calculating the Fourier transforms $H(\omega)$ and $G(\omega)$, multiply them to get $F(\omega)$, and we get $f(x)$ by taking the inverse Fourier transform of $F(\omega)$. Often this is a much faster method then calculating the convolution integral, because the routines that calculate the Fourier transform are so fast. We look at an example where we filter noise from the data. We simulate a signal of a noisy ECG recording by some sine functions:

```
data = Table[
   N[Sin[2 π/256 2 t] + Sin[2 π/256 6 t] + 0.75 (Random[ ] - 1 / 2)], {t, 256} ] ;
ListPlot[data, AspectRatio -> .2, PlotJoined -> True,
  ImageSize -> 400, AxesLabel -> {"time", "Ampl"}];
```

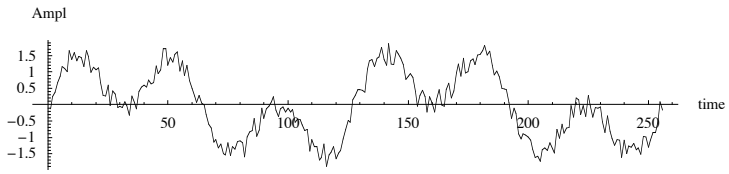

Figure B.6 A discrete input signal with additive uncorrelated uniform noise.

We make a filter with a Gaussian shape and shift it to the origin:

```
σ = 5.; kernel = Table[gauss[x, σ], {x, -128, 127}];
kernel = RotateLeft[kernel, 128];
ListPlot[kernel, PlotRange -> {{0, 256}, All}, PlotJoined -> True,
  AspectRatio -> .2, AxesLabel -> {"Freq", "Ampl"}, ImageSize -> 400];
```

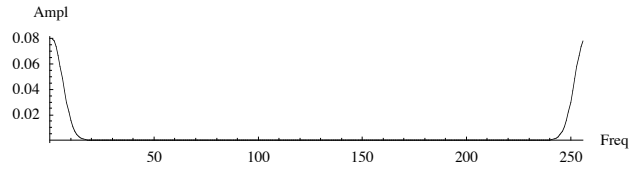

Figure B.7 The Gaussian kernel in the spatial domain, shifted to the origin. The right half of the frequency axis (i.e. 129-256) is often called the negative frequency axis.

This is a low-pass filter. We now filter (convolve) in the Fourier domain (note that a space denotes multiplication in *Mathematica*):

```
conv = Sqrt[256] InverseFourier[Fourier[data] Fourier[kernel]];
ListPlot[ conv, AspectRatio -> .2, PlotJoined -> True,
  AxesLabel -> {"time", "Ampl"}, ImageSize -> 400];
```

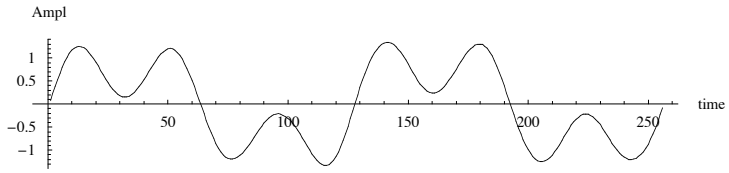

Figure B.8 The filtered result.

As expected, the noise did not pass the filter.