Filters and Fourier Transforms

NOTE:

Before reading these notes, see 15-462 “Basic Raster” notes:

http://www.cs.cmu.edu/afs/cs/academic/class/15462/web/notes/notes.html

OUTLINE:

The Cost of Filtering

Fourier Transforms

Properties of Convolution

both continuous and discrete convolution are

commutative: $a \quad h = h \quad a$

so the distinction between signal (a) and filter impulse response (h) is blurred

associative: $a \quad (h_1 \quad h_2) = (a \quad h_1) \quad h_2$

so instead of convolving signal a with complicated filter h, if h can be written as $h=h_1 \quad h_2$, then a can be filtered in two passes: first with h_1 and then with h_2

Optimizing Filtering

Filtering can be slow.

- If the impulse response has a *support* (width of nonzero portion) of $S_x \times S_y$ pixels, then the cost of filtering is $O(S_x S_y)$ per output pixel. Filters with large support are expensive, in general. Filtering an N×N picture with an S×S impulse response costs $O(N^2 S^2)$ using standard formula - exorbitant!

Certain filtering operations can be optimized.

- Separable Filters
  - A filter h that can be written in the form $h[x,y]=h_x(x)\ h_y(y)$ is said to be *separable*. If the supports of h_x and h_y are S_x and S_y , then computing a   h directly costs $O(S_x S_y)$ per output pixel, but exploiting separability and associativity, we can do two-pass filtering, $(a \quad h_x) \quad h_y$, with cost of only $O(S_x+S_y)$.
- Box Filters
  - A 1-D box filter of width S can be computed in O(1) time per output pixel.
  - A 2-D box filter of size S×S can be computed in O(1) time also.

Fourier convolution:

- optimizes general 2-D convolution to $O(N^2 \log N)$, as we will see later.

Fast Box Filtering

1-D box filtering can be done in O(1) time per output pixel

A 1-D box filter of width S=2K+1 is: $b[x] = \frac{1}{S} \sum_{t=x-K}^{x+K} a[t]$

With this formula, cost is O(S) -- slow for wide filters

But note that b[x+1]-b[x] = (a[x+K+1]-a[x-K])/S, so if we compute incrementally, adding in at the leading edge of the filter window, and subtracting out at the trailing edge, we get this fast algorithm:

```
initialize b
for x
    output b
    b += (a[x+K+1]-a[x-K])/S
```

2-D box filtering can also be done in constant time per output pixel

Do you see how to generalize the 1-D add-in/subtract-out trick to 2-D?

I know of three ways to do this. (This comes up again for texture mapping).

Cost of Filtering Algorithms

ALGORITHM	1-D signal length = N filter width = S	2-D picture size = N×N filter size = S×S
straightforward	$O(NS)$	$O(N^2S^2)$
box filter	$O(N)$	$O(N^2)$
separable filter	N.A.	$O(N^2S)$
Fourier convolution with FFT	$O(N\log N)$	$O(N^2\log N)$

Frequency Domain

We can visualize & analyze a signal or a filter in either the spatial domain or the frequency domain.

Spatial domain: x, distance (usually in pixels).

Frequency domain: can be measured with either:

ω, **angular frequency** in radians per unit distance, or

f, **rotational frequency** in cycles per unit distance. $\omega = 2\pi f$.

We'll use ω mostly.

The **period** of a signal, $T = 1/f = 2\pi/\omega$.

Examples:

The signal [0 1 0 1 0 1 ...] has frequency $f = .5$ (.5 cycles per sample).

The signal [0 0 1 1 0 0 1 1 ...] has frequency $f = .25$.

Fourier Transform

The **Fourier transform** is used to transform between the spatial domain and the frequency domain. A **transform pair** is symbolized with "↔", e.g. $f \leftrightarrow F$.

SPATIAL DOMAIN ↔ FREQUENCY DOMAIN

signal $f(x)$ ↔ **spectrum** $F(\omega)$

Fourier Transform : $$F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

Inverse Fourier Transform : $$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega$$

where $i = \sqrt{-1}$. Note that F will be complex, in general.

Filtering Terminology

For a linear, shift-invariant filter,

$$\begin{tikzcd} \text{input signal} \arrow[r] & \boxed{\text{FILTER}} \arrow[r] & \text{output signal} \end{tikzcd}$$

A filter can be described in the spatial domain by its **impulse response**† $h(x)$, its response to a delta function input, as a function of position. Abbrev: IR.

$\delta(x) \rightarrow \text{FILTER} \rightarrow h(x)$

† a.k.a. **point spread function** in image processing

And it can be described in the frequency domain by its **frequency response** $H(\omega)$, its response to a sinusoid input as a function of frequency. Abbrev: FR.

$\sin(\omega x) \rightarrow \text{FILTER} \rightarrow \mathrm{H}(\omega)\sin(\omega x)$

$\mathrm{H}(\omega)$ is the **gain** of the filter at frequency ω.

The FR is the Fourier transform of the IR: $h(x) \leftrightarrow \mathrm{H}(\omega)$.

Note the terminology distinction. For a signal: signal $\leftrightarrow$ spectrum,

but for a filter: impulse response $\leftrightarrow$ frequency response.