Antialiasing

and Related Issues

OUTLINE:

Antialiasing

Prefiltering, Supersampling, Stochastic Sampling

Rastering and Reconstruction

Gamma Correction

Antialiasing Methods

To reduce aliasing, either:

1. fix the signal by **prefiltering**:
   Reduce the signal's bandwidth by low pass filtering before sampling.
   Highest quality method, but often impractical.
2. fix the samples by **supersampling**:
   Use more samples to raise the Nyquist frequency.
   Simple and widely used.
3. fix the samples by **stochastic sampling**:
   Sample randomly, not uniformly.
   Relatively simple, usually used in combination with supersampling.

In practice, we rarely eliminate aliasing entirely, we merely reduce it to tolerable levels.

Filtering Terminology

For a linear, shift-invariant filter, input signal → FILTER → output signal

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.

$\delta(x) \rightarrow$ 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.

$\sin(\omega x) \rightarrow$ FILTER $\rightarrow H(\omega)\sin(\omega x)$

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

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

Note the terminology distinction. For a signal: signal↔spectrum,

but for a filter: impulse response↔frequency response.

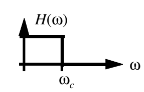
Low Pass Filters

A **low pass filter** passes (preserves) low frequencies while stopping (eliminating) high frequencies.

The **ideal low pass filter** has gain 1 at frequencies below the cutoff, and gain 0 above the cutoff ω_c.

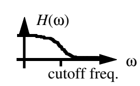
Its IR is a sinc, its FR is a box:

$(\omega_c/\pi)\mathrm{sinc}((\omega_c/\pi)x) \leftrightarrow \mathrm{box}(\omega/2\omega_c)$



FR of an ideal low pass filter

In practice, you typically don't want the ideal low pass filter, since it has infinite **support** (width), and it causes ripples in output. Instead, we use finite impulse response filters that attenuate, but do not stop high frequencies. The choice of filter is usually governed by a speed/quality tradeoff.



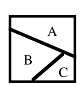
FR of a typical low pass filter

Method 1: Prefiltering

The idea: **Antialias by low pass filtering the signal, bandlimiting it to the Nyquist frequency before sampling.**

An approximation to an ideal low pass filter (sinc) is typically used.

Example: antialiasing polygons with box filtering:
find the fraction of the pixel covered by each polygon,
compute weighted average of colors.



Prefiltering is easy and widely used when the signal is discrete, as when resampling.

It's harder when the signal is continuous. Prefiltering is difficult in 3-D rendering when visibility is not known exactly. It is more practical for simple signals, e.g. signals given by a closed form expression or 2-D text and graphics (lines, polygons, fonts).

Method 2: Supersampling

The idea: **Antialias by increasing the sampling frequency**.

Example: render at k times the resolution. Then, if your display supports the higher resolution (e.g. film) then use it, otherwise (e.g. video) downsample by factor of k to the display resolution.

Practical constraints:

Sampling at k times the resolution, in 2-D, will typically cost k^2 times more in time and perhaps in memory as well.

To eliminate the memory problem, you can usually downsample on the fly.

Theoretical problems:

How do you know how much to supersample? i.e. how do you pick k?

If one value of k works everywhere, then do **uniform supersampling**.

If not, use **adaptive supersampling**: take several samples within pixel; if their variance is low, return their average, otherwise subdivide pixel and recurse, like a quadtree.

Method 3: Stochastic Sampling

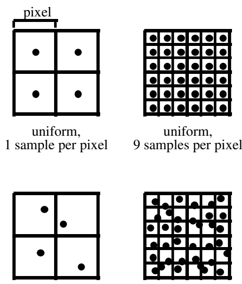
Instead of sampling on a uniform grid, sample at random points.

Several ways to do this:

1. **jittering**: choose a point at random, uniformly within the domain (interval, pixel area, or frame time). *Easy.*
2. **Poisson disk sampling**: pick a point, keep if its nearest neighbor is >r units away, discard if not. Repeat until full. *Harder, better.*

Stochastic sampling is typically used in combination with supersampling. Common: 16 samples per pixel.

Stochastic sampling reduces aliasing, but increases noise; it's not a panacea.



uniform, 1 sample per pixel | uniform, 9 samples per pixel | jittered, 1 sample per pixel | jittered, 9 samples per pixel

Resampling

Resampling: sampling a discrete signal at a different sampling rate.

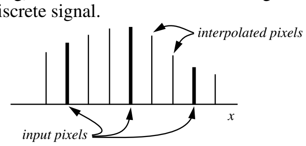
Example: "zooming" a picture from n_x by n_y pixels to sn_x by sn_y pixels

- s=1: no change
- s>1: called **upsampling** or **interpolation**
  - can lead to blocky appearance if point sampling is used, since pixels are magnified
  - This artifact is **rastering**, which is related to aliasing, but different.
    - *cure is better reconstruction of contin. signal.*
    - *Easy.*
- s<1: called **downsampling** or **decimation**
  - can lead to moire patterns and jaggies
  - This artifact is **aliasing,**
    - *cure is sampling at a higher frequency, or low pass filtering before sampling.*
    - *Harder.*

Rastering and Reconstruction

When upsampling (interpolating), we want to reconstruct the original continuous signal from a discrete signal.

rastering results when a poor reconstruction filter, such as a box filter, is used.



box filter → pixel replication, blocky appearance

triangle filter → linear interpolation, better appearance.

In 2-D, use of a triangle filter is called **bilinear interpolation**.

More expensive cubic filters are sometimes used, but bilinear interpolation is often sufficient. See Mitchell & Netravali, SIGGRAPH 88 for details on good cubic, support 4 reconstruction filters.

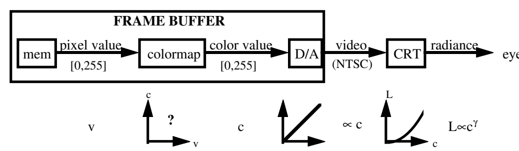
Most Displays have Nonlinear Intensity

Printing is nonlinear: 

darkness of 2 black dots less than twice the darkness of 1 bla

Cathode ray tubes are nonlinear:

color 128 is darker than the average of colors 0 and 255



Brightness of light on CRT has units of radiance (a.k.a. "intensity").

The **gamma** $\gamma \in [2,3]$, typically.

Correcting for this nonlinearity is **gamma correction.**

Gamma Correction

Video cameras compensate by generating $c \propto L^{1/\gamma}$

"gamma corrected video"

When filtering, we want to work in units of radiance L (not $L^{1/\gamma}$)

because eye's lens does linear filtering of light, and radiance is the natural unit for measuring light

Gamma correction is critical in order to do good antialiasing!

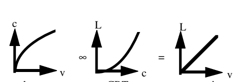
To determine the gamma of your monitor:

display this:  adjust gamma until L & R appear equally bright

On SGI, **"`gamma x`"** sets colormap: $c[v] = 255(v/255)^{1/x}$

Two Methods for Gamma Correction

1. HARDWARE: build compensation into colormap



colormap: $c[v] \propto v^{1/\gamma}$

so $v \propto L$

problem: visible quantization of dark pixels

2. SOFTWARE: do compensation during pixel I/O

use colormap: c[v] = v

– after reading a pixel with value v, convert to $L \propto v^{\gamma}$

– before writing a pixel, convert from radiance L to $v \propto L^{1/\gamma}$

8 bits for v ⇒ 16-20 bits for L

(better than c[v]=v, quantization-wise, is colormap that's slightly concave upward)

Antialiasing Advice

Try point sampling first. If that's not good enough:

When reconstructing or resampling a signal, use (bi)linear interpolation or better.

When sampling a continuous signal, as in rendering, do prefiltering if you can, otherwise do supersampling, e.g. 16 samples per pixel.

Stochastic supersampling (e.g. jittering) is better than uniform supersampling.

Pay close attention to the gamma of your display.

If you're creating animation, determine if motion blur (temporal antialiasing is necessary), and be extra-careful about spatial antialiasing.