11. Sampling theory
Sampled representations

- How to store and compute with continuous functions?
- Common scheme for representation: samples
  write down the function’s values at many points
Reconstruction

- Making samples back into a continuous function
  for output (need realizable method)
  for analysis or processing (need mathematical method)
  amounts to “guessing” what the function did in between
Filtering

- Processing done on a function can be executed in continuous form (e.g. analog circuit) but can also be executed using sampled representation.
- Simple example: smoothing by averaging.
Roots of sampling

• Nyquist 1928; Shannon 1949
  famous results in information theory

• 1940s: first practical uses in telecommunications

• 1960s: first digital audio systems

• 1970s: commercialization of digital audio

• 1982: introduction of the Compact Disc
  the first high-profile consumer application

• This is why all the terminology has a communications or audio “flavor”
  early applications are 1D; for us 2D (images) is important
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  how can we be sure we are filling in the gaps correctly?
Undersampling

• What if we “missed” things between the samples?

• Simple example: undersampling a sine wave
  
  unsurprising result: information is lost
  
  surprising result: indistinguishable from lower frequency
  
  also was always indistinguishable from higher frequencies

  aliasing: signals “traveling in disguise” as other frequencies
Undersampling

• What if we “missed” things between the samples?
• Simple example: undersampling a sine wave
  unsurprising result: information is lost
  surprising result: indistinguishable from lower frequency
  also was always indistinguishable from higher frequencies

*aliasing*: signals “traveling in disguise” as other frequencies
Undersampling

- What if we “missed” things between the samples?

- Simple example: undersampling a sine wave
  
  unsurprising result: information is lost
  
  surprising result: indistinguishable from lower frequency
  
  also was always indistinguishable from higher frequencies

  *aliasing*: signals “traveling in disguise” as other frequencies
Undersampling

• What if we “missed” things between the samples?
• Simple example: undersampling a sine wave
  unsurprising result: information is lost
  surprising result: indistinguishable from lower frequency
  also was always indistinguishable from higher frequencies
  aliasing: signals “traveling in disguise” as other frequencies
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  unsurprising result: information is lost
  surprising result: indistinguishable from lower frequency
  also was always indistinguishable from higher frequencies
  aliasing: signals “traveling in disguise” as other frequencies
Undersampling

• What if we “missed” things between the samples?
• Simple example: undersampling a sine wave
  unsurprising result: information is lost
  surprising result: indistinguishable from lower frequency
  also was always indistinguishable from higher frequencies
  aliasing: signals “traveling in disguise” as other frequencies
Sneak preview

• Sampling creates copies of the signal at higher frequencies
• Aliasing is these frequencies leaking into the reconstructed signal
  – frequency $f_s - x$ shows up as frequency $x$
• The solution is filtering
  – during sampling, filter to keep the high frequencies out so they don’t create aliases at the lower frequencies
  – during reconstruction, again filter high frequencies to avoid including high-frequency aliases in the output.
Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)
Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)
Linear filtering: a key idea

• Transformations on signals; e.g.:
  bass/treble controls on stereo
  blurring/sharpening operations in image editing
  smoothing/noise reduction in tracking

• Key properties
  linearity: \( \text{filter}(f + g) = \text{filter}(f) + \text{filter}(g) \)
  shift invariance: behavior invariant to shifting the input
  • delaying an audio signal
  • sliding an image around

• Can be modeled mathematically by \textit{convolution}
Convolution warm-up

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing

![Convolution Example](image)
Convolution warm-up

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing
Convolution warm-up

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing
Convolution warm-up

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing
Convolution warm-up

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing
Convolution warm-up

- Same moving average operation, expressed mathematically:

\[ b_{\text{smooth}}[i] = \frac{1}{2r + 1} \sum_{j=i-r}^{i+r} b[j] \]
Discrete convolution

• Simple averaging:

\[ b_{\text{smooth}}[i] = \frac{1}{2r + 1} \sum_{j=i-r}^{i+r} b[j] \]

every sample gets the same weight

• Convolution: same idea but with \textit{weighted} average

\[ (a \ast b)[i] = \sum_{j} a[j]b[i-j] \]

each sample gets its own weight (normally zero far away)

• This is all convolution is: it is a \textbf{moving weighted average}
Filters

• Sequence of weights $a[j]$ is called a filter

• Filter is nonzero over its region of support
  usually centered on zero: support radius $r$

• Filter is normalized so that it sums to 1.0
  this makes for a weighted average, not just any old weighted sum

• Most filters are symmetric about 0
  since for images we usually want to treat left and right the same

$a$ box filter

\[ \frac{1}{2r+1} \]
Convolution and filtering

- Can express sliding average as convolution with a box filter
- \( a_{\text{box}} = [..., 0, 1, 1, 1, 1, 1, 0, ...] \)
Example: box and step

\[ b[i] \]

\[ a[j] \]
Example: box and step

\[ b[i] \]

\[ a[j] \]

\[ 1 \]

\[ 0 \]

\[ i \rightarrow \]

\[ \times \]

\[ = \]

\[ a[j] \times b[i-j] = 0 0 0 0 0 0 0 0 \]
Example: box and step

\[ b[i] \times a[j] = \sum_{j} a[j] b[i-j] \]

\[ (a \ast b)[i] \]

\[ 0 \quad -7 \]

\[ 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \]

\[ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \]
Example: box and step

\[
\begin{align*}
\sum a[j] b[i-j] & = 0 0 0 0 0 0 0 0 \\
(a * b)[i] & = 0 0 .2 .2 .2 0 \\
\end{align*}
\]
Example: box and step
Convolution and filtering

- Convolution applies with any sequence of weights
- Example: bell curve (gaussian-like) \([..., 1, 4, 6, 4, 1, ...]/16\)
Convolution and filtering

- Convolution applies with any sequence of weights.
- Example: bell curve (gaussian-like) $[\ldots, 1, 4, 6, 4, 1, \ldots]/16$
Discrete convolution

- Notation: \( b = c \ast a \)

- Convolution is a multiplication-like operation
  
  - commutative: \( a \ast b = b \ast a \)
  
  - associative: \( a \ast (b \ast c) = (a \ast b) \ast c \)
  
  - distributes over addition: \( a \ast (b + c) = a \ast b + a \ast c \)
  
  - scalars factor out: \( \alpha a \ast b = a \ast \alpha b = \alpha (a \ast b) \)

  - identity: unit impulse \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \)
    
    \[ a \ast e = a \]

- Conceptually no distinction between filter and signal
Discrete filtering in 2D

- Same equation, one more index

\[(a \star b)[i, j] = \sum_{i', j'} a[i', j'] b[i - i', j - j']\]

now the filter is a rectangle you slide around over a grid of numbers

- Commonly applied to images
  - blurring (using box, using gaussian, …)
  - sharpening (impulse minus blur)

- Usefulness of associativity
  - often apply several filters one after another: \(((a \star b_1) \star b_2) \star b_3\)
  
  this is equivalent to applying one filter: \(a \star (b_1 \star b_2 \star b_3)\)
Continuous convolution: warm-up

- Can apply sliding-window average to a continuous function just as well
  output is continuous
  integration replaces summation
Continuous convolution: warm-up

- Can apply sliding-window average to a continuous function just as well
  output is continuous
  integration replaces summation
Continuous convolution: warm-up

- Can apply sliding-window average to a continuous function just as well
  output is continuous
  integration replaces summation
Continuous convolution: warm-up

- Can apply sliding-window average to a continuous function just as well
  output is continuous
  integration replaces summation
Continuous convolution: warm-up

- Can apply sliding-window average to a continuous function just as well
  output is continuous
  integration replaces summation
Continuous convolution

- Sliding average expressed mathematically:

\[ g_{\text{smooth}}(x) = \frac{1}{2r} \int_{x-r}^{x+r} g(t) \, dt \]

Note difference in normalization (only for box)

- Convolution just adds weights

\[ (f \ast g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t) \, dt \]

Weighting is now by a function

Weighted integral is like weighted average

Again bounds are set by support of \( f(x) \)
One more convolution

- Continuous–discrete convolution

\[(a \ast f)(x) = \sum_{i} a[i] f(x - i)\]

\[(a \ast f)(x, y) = \sum_{i, j} a[i, j] f(x - i, y - j)\]

used for reconstruction and resampling
Continuous-discrete convolution
Continuous-discrete convolution
Continuous-discrete convolution

Diagram showing samples and a reconstructed signal with a summation symbol (\(\Sigma\)) indicating the convolution process.
Continuous-discrete convolution

Diagram showing a continuous signal with discrete samples and a reconstruction process.
Continuous-discrete convolution

samples

reconstructed signal
Resampling

• Changing the sample rate
  in images, this is enlarging and reducing

• Creating more samples:
  increasing the sample rate
  “upsampling”
  “enlarging”

• Ending up with fewer samples:
  decreasing the sample rate
  “downsampling”
  “reducing”
Resampling

- Reconstruction creates a continuous function
  forget its origins, go ahead and sample it
Resampling

- Reconstruction creates a continuous function
  forget its origins, go ahead and sample it
Resampling

- Reconstruction creates a continuous function
  forget its origins, go ahead and sample it
Resampling

- Reconstruction creates a continuous function forget its origins, go ahead and sample it
Resampling

- Reconstruction creates a continuous function forget its origins, go ahead and sample it
Cont.–disc. convolution in 2D

- same convolution—just two variables now

\[(a \ast f)(x, y) = \sum_{i,j} a[i, j] f(x - i, y - j)\]

- loop over nearby pixels, average using filter weight
- looks like discrete filter, but offsets are not integers and filter is continuous
- remember placement of filter relative to grid is variable
A gallery of filters

- Box filter
  Simple and cheap

- Tent filter
  Linear interpolation

- Gaussian filter
  Very smooth antialiasing filter

- B-spline cubic
  Very smooth

- Catmull-rom cubic
  Interpolating

- Mitchell-Netravali cubic
  Good for image upsampling
Box filter

\[ a_{\text{box},r}[i] = \begin{cases} 
\frac{1}{2r + 1} & |i| \leq r, \\
0 & \text{otherwise}. 
\end{cases} \]

\[ f_{\text{box},r}(x) = \begin{cases} 
\frac{1}{2r} & -r \leq x < r, \\
0 & \text{otherwise}. 
\end{cases} \]
Tent filter

$$f_{\text{tent}}(x) = \begin{cases} 
1 - |x| & |x| < 1, \\
0 & \text{otherwise}; 
\end{cases}$$

$$f_{\text{tent}, r}(x) = \frac{f_{\text{tent}}(x/r)}{r}.$$
Gaussian filter

\[ f_g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \]
B-Spline cubic

\[ f_B(x) = \frac{1}{6} \begin{cases} 
-3(1 - |x|)^3 + 3(1 - |x|)^2 + 3(1 - |x|) + 1 & -1 \leq x \leq 1, \\
(2 - |x|)^3 & 1 \leq |x| \leq 2, \\
0 & \text{otherwise}.
\]
Catmull-Rom cubic

\[ f_C(x) = \frac{1}{2} \begin{cases} 
-3(1 - |x|)^3 + 4(1 - |x|)^2 + (1 - |x|) & -1 \leq x \leq 1, \\
(2 - |x|)^3 - (2 - |x|)^2 & 1 \leq |x| \leq 2, \\
0 & \text{otherwise}. 
\end{cases} \]
Michell-Netravali cubic

\( f_M(x) = \frac{1}{3} f_B(x) + \frac{2}{3} f_C(x) \)

\[
= \frac{1}{18} \begin{cases} 
-21(1 - |x|)^3 + 27(1 - |x|)^2 + 9(1 - |x|) + 1 & -1 \leq x \leq 1, \\
7(2 - |x|)^3 - 6(2 - |x|)^2 & 1 \leq |x| \leq 2, \\
0 & \text{otherwise.}
\end{cases}
\]
Effects of reconstruction filters

• For some filters, the reconstruction process winds up implementing a simple algorithm

• Box filter (radius 0.5): nearest neighbor sampling
  
  box always catches exactly one input point
  
  it is the input point nearest the output point
  
  so output[i, j] = input[round(x(i)), round(y(j))]
  
  x(i) computes the position of the output coordinate i on the input grid

• Tent filter (radius 1): linear interpolation
  
  tent catches exactly 2 input points
  
  weights are a and (1 – a)
  
  result is straight-line interpolation from one point to the next
Properties of filters

- Degree of continuity
- Impulse response
- Interpolating or no
- Ringing, or overshoot

interpolating filter used for reconstruction
Ringing, overshoot, ripples

- **Overshoot**
  caused by negative filter values

- **Ripples**
  constant in, non-const. out
  ripple free when:

\[
\sum_i f(x + i) = 1 \text{ for all } x.
\]
Constructing 2D filters

• Separable filters (most common approach)
Reducing and enlarging

• Very common operation
devices have differing resolutions
applications have different memory/quality tradeoffs

• Also very commonly done poorly

• Simple approach: drop/replicate pixels

• Correct approach: use resampling
Reducing and enlarging

• Very common operation
  devices have differing resolutions
  applications have different memory/quality tradeoffs
• Also very commonly done poorly
• Simple approach: drop/replicate pixels
• Correct approach: use resampling
by dropping pixels  

250 pixel width  

gaussian filter
box reconstruction filter

bicubic reconstruction filter

4000 pixel width
Types of artifacts

• Garden variety
  what we saw in this natural image
  fine features become jagged or sparkle

• Moiré patterns
600ppi scan of a color halftone image
downsampling a high resolution scan

by dropping pixels

 gaussian filter
Types of artifacts

• Garden variety
  what we saw in this natural image
  fine features become jagged or sparkle

• Moiré patterns
  caused by repetitive patterns in input
  produce large-scale artifacts; highly visible

• These artifacts are aliasing just like in the audio example earlier

• How do I know what filter is best at preventing aliasing?
  practical answer: experience
  theoretical answer: there is another layer of cool math behind all this
  • based on Fourier transforms
  • provides much insight into aliasing, filtering, sampling, and reconstruction
Checkpoint

• Want to formalize sampling and reconstruction
  – define impulses
  – then we can talk about S&R with only one datatype
• Define Fourier transform
• Destination: explaining how aliases leak into result
Mathematical model

• We have said sampling is storing the values on a grid
• For analysis it’s useful to think of the sampled representation in the same space as the original
  – I’ll do this using *impulse functions* at the sample points
Impulse function

• A function that is confined to a very small interval
  – but still has unit integral
  – really, the limit of a sequence of ever taller and narrower functions
  – also called Dirac delta function

• Key property: multiplying by an impulse selects the value at a point
  – Defn via integral

• Impulse is the identity for convolution
  – “impulse response” of a filter
Sampling & recon. reinterpreted

- Start with a continuous signal
- Convolve it with the sampling filter
- Multiply it by an impulse grid
- Convolve it with the reconstruction filter
Checkpoint

• Formalized sampling and reconstruction
  – used impulses with multiplication and convolution
  – can talk about S&R with only one datatype
• Define Fourier transform
• Destination: explaining how aliases leak into result
Fourier series

- Probably familiar idea of adding up sines and cosines to approximate a periodic function
Fourier series
Fourier transform

• Like Fourier series but for aperiodic functions
  – Fourier series: only multiples of base frequency
• Fourier transform: let period go to infinity
  – eventually all frequencies are needed
  – result: countable sum turns into integral
The Fourier transform

• Any function on the real line can be represented as an infinite sum of sine waves
The Fourier transform
The Fourier transform

- The coefficients of those sine waves form a continuous function of frequency
- That function, which has the same datatype as the first one, is the Fourier transform.

\[ F(u) = \int_{-\infty}^{\infty} f(x)(\cos 2\pi ux - i \sin 2\pi ux)dx \]

- Phase encoded in complex number
Fourier transform properties

- F.T. is its own inverse (just about)
- Frequency space is a dual representation
  - amplitude known as “spectrum”
Fourier pairs

- sinusoid — impulse pair
- box — sinc
- tent — sinc^2
- bspline — sinc^4
- gaussian — gaussian (inv. width)
- imp. grid — imp. grid (1/d spacing)
Fourier pairs

- sinusoid — impulse pair
- box — sinc
- tent — sinc²
- bspline — sinc⁴
- gaussian — gaussian (inv. width)
- imp. grid — imp. grid (1/d spacing)
Fourier pairs

- sinusoid — impulse pair
- box — sinc
- tent — sinc²
- bspline — sinc⁴
- gaussian — gaussian (inv. width)
- imp. grid — imp. grid (1/d spacing)
More Fourier facts

• F.T. preserves energy
  – That is, the squared integral
• DC component (average value)
  – It shows up at F(0)
More Fourier facts

• Dilation (stretching/squashing)
  – Results in inverse dilation in F.T.
Convolution and multiplication

- They are dual to one another under F.T.

\[ \mathcal{F}\{f \ast g\}(u) = F(u)G(u) \]
\[ \mathcal{F}\{fg\}(u) = (F \ast G)(u) \]

- Lowpass filters
  - Most of our “blurring” filters have most of their F.T. at low frequencies
  - Therefore they attenuate higher frequencies
Checkpoint

• Formalized sampling and reconstruction
  – used impulses with multiplication and convolution
• Can talk about S&R with only one datatype
• Defined Fourier transform
  – alternate representation for functions
  – turns convolution, which seems hard, into multiplication, which is easy
• Destination: explaining how aliases leak into result
Sampling and reconstruction in F.T.

• Look at our sampling/reconstruction formulation in Fourier domain
  – Convolve with filter = remove high frequencies
  – Multiply by impulse grid = convolve with impulse grid
    • that is, make a bunch of copies
  – Convolve with filter = remove extra copies
  – Left with approximation of original
    • but filtered a couple of times
Aliasing in sampling/reconstruction
Aliasing in sampling

- If sampling filter is not adequate, spectra will overlap
- No way to fix once it’s happened
  - can only use drastic reconstruction filter to eliminate
- Nyquist criterion
Preventing aliasing in sampling

• Use high enough sample frequency
  – works when signal is *band limited*
  – sample rate $2 \times$ (highest freq.) is enough to capture all details

• Filter signal to remove high frequencies
  – make the signal band limited
  – remove frequencies above $0.5 \times$ (sample freq.) (Nyquist)
Effect of sample rate on aliasing

- Original image
- Sampled image
- Sampled × 2 image
- Sampled × 4 image

Graphs showing the effect of different sample rates on the frequency spectrum, with aliasing becoming more pronounced as the sample rate decreases.
Smoothing (lowpass filtering)
Effect of smoothing on aliasing

original

samp.: no filter

samp.: mild blur

samp.: strong blur

severe aliasing

some aliasing

minimal aliasing
Aliasing in reconstruction

- If reconstruction filter is inadequate, will catch alias spectra
- Result: high frequency alias components
- Can happen even if sampling is ideal
Aliasing in reconstruction

- If reconstruction filter is inadequate, will catch alias spectra
- Result: high frequency alias components
- Can happen even if sampling is ideal
Reconstruction filters
Sampling filters

- “Ideal” is box filter in frequency
  - which is sinc function in space
- Finite support is desirable
  - compromises are necessary
- Filter design: passband, stopband, and in between

\[ G_{\text{filter}}. \]
Useful sampling filters

• Sampling theory gives criteria for choosing
• Box filter
  – sampling: unweighted area average
  – reconstruction: e.g. LCD
• Gaussian filter
  – sampling: gaussian-weighted area average
  – reconstruction: e.g. CRT
• Piecewise cubic
  – good small-support reconstruction filter
  – popular choice for high-quality resampling (next lecture)
Resampling filters

- Resampling, logically, is two steps
  - first: reconstruct continuous signal
  - second: sample signal at the new sample rate
- Each step requires filtering
  - reconstruction filter
  - sampling filter
- This amounts to two successive convolutions
  - so regroup into one operation:
    \[ f_{\text{samp}} \ast f_{\text{recon}} \ast g = (f_{\text{samp}} \ast f_{\text{recon}}) \ast g \]
  - single filter both reconstructs and antialiases
Resampling in frequency space
Sizing reconstruction filters

• Has to perform as a reconstruction filter
  – has to be at least big enough relative to input grid
• Has to perform as a sampling filter
  – has to be at least big enough relative to output grid
• Result: filter size is max of two grid spacings
  – upsampling (enlargement): determined by input
  – downsampling (reduction): determined by output
  – for intuition think of extreme case (10x larger or smaller)
How this plays out in n-D

• Fourier transform is in terms of “plane waves”

\[ F(u) = \int_{\mathbb{R}^n} f(x) e^{-2\pi i x \cdot u} \, dx \]

• Separable products of 1D functions transform separably

\[ \mathcal{F}\{f(x)g(y)\} = F(u)G(v) \]
How this plays out in n-D

- By separability everything goes through the same as in 1D
  - impulse grid, filter, reconstruction
- Possibility of non-rectangular band-limiting
  - any region that does not overlap is fair game

\[ g \times \hat{k} = \hat{g} \hat{k} \]
Sampling in n-D

• With sampling on a regular lattice and reconstruction with a separable filter, everything is pretty much the same
• Non-rectangular grids are possible
  – Hexagonal arrays in 2D
  – FCC and BCC grids for volume data
  – Interlaced video
• Band limiting now means an n-D region
  – cubes are fine
  – anything that is non-overlapping is also fine
Summary

• Want to explain aliasing and answer questions about how to avoid it
• Formalized sampling and reconstruction using impulse grids and convolution
• Fourier transform gives insight into what happens when we sample
• Nyquist criterion tells us what kind of filters to use