PBRT: Sampling and Reconstruction

Wednesday, December 16, 2020 10:58 AM

7 Sampling and reconstruction

An RGB to SPD mapping is used by a display to obtain the spectral power distribution of each ***image pixel*** that the corresponding ***display pixels*** will emit.

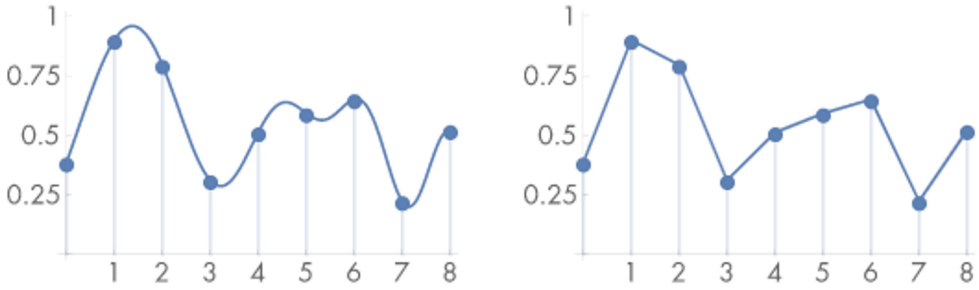
7.1 Sampling theory

Sampling is the process of taking discrete values of a continuous function. ***Reconstruction*** is the process of converting the discrete samples back to a continuous function, which tries to approximate the original one as much as possible.

Incident radiance is a continuous function defined over the film plane and that we try to approximate by sampling the values of discrete pixels. The pixel image is a discretization of this function.

Sampling is performed by ***tracing rays*** and the result is a digital image. Reconstruction is performed by the ***display***, typically via some form of interpolation. The function is defined at all points on the display.

Sampling of a 1D function
Values x' of the domain are called ***sample points*** and $f('x)$s ***sample values***
Sample points are ***equally spaced***
The reconstructed function on the right (a piecewise linear function) is denoted $\tilde{f}$ and was obtained by linearly interpolating the sample values



We don't use linear or polynomial approximation because these use the derivative of the function and we don't know the function, much less its derivative.

To understand sampling and reconstruction, it's crucial to understand what a ***pixel*** really is: A pixel is just a point with no dimension which has been assigned a color or blend of colors. An image is represented in storage as a rectangular array of color values. An image is displayed by a device by reconstructing the radiance of the real-world or virtual scene of which the pixels are just samples. Reconstruction is done with a ***filter***, which is usually some form of interpolation: linear, bicubic, windowed sinc function, nearest-neighbor, truncated Gaussian, box.
Ray-traced images that have aliasing were sampled at regularly-spaced sample points.

PBRT's ***Samplers*** leverage ***low-discrepancy point sets***, which are a particular type of well-distributed sample points. All the samplers generate n-dimensional sample vectors.

7.1.1 The frequency domain and the Fourier transform

Fourier analysis evaluates the quality of the match between the reconstructed function and the original.

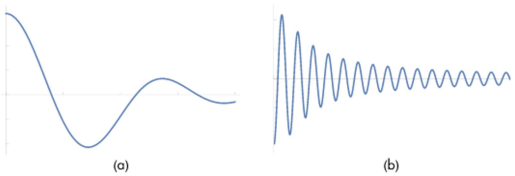
Functions are normally expressed in the ***spatial domain***, but the Fourier transform represents a function in the ***frequency domain***.

Functions exist both in spatial space/domain and in ***frequency space/domain***.

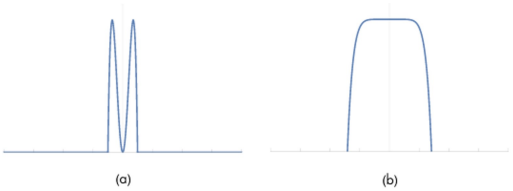
A frequency-domain graph shows how much of the signal lies within each frequency.

Functions that vary slowly are said to have ***low-frequency content***. Functions that vary rapidly have ***high-frequency content***.

Functions plotted over the spatial domain



Functions plotted over the frequency domain
*These are referred to as the **spectra** of their respective spatial-domain functions*



Most functions are ***linear combinations*** of ***shifted sinusoid basis functions*** in ***frequency space***. The Fourier transform takes a function and maps it to its corresponding linear combination of basis sinusoid functions.

This frequency-domain representation allows us to see the contribution of each frequency ω to the value of the function: the function's distribution of frequencies.

Given a frequency-domain representation, Fourier analysis is used to gain insight into the error that is introduced by sampling and reconstruction.

The Fourier transform (or ***analysis equation***) of a function $f(x)$ is:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\omega x} dx$$

This is not a Riemannian definite integral, because the interval of integration is not finite. It can't be evaluated using the Fundamental Theorem of Calculus.

$$\int_{-\infty}^{\infty} f(x) dx$$

is an ***improper integral*** and is shorthand for:

$$\lim_{a\to-\infty}\int_a^c f(x)dx + \lim_{b\to\infty}\int_c^b f(x)dx$$

Its inverse (or ***synthesis equation***) is:

$$f(x) = \int_{-\infty}^{\infty} F(\omega)e^{i2\pi\omega x}d\omega$$

which maps back to the spatial domain.

These are the Fourier pairs of the most common functions used in sampling and reconstruction:

Spatial Domain	Frequency Space Representation
Box: $f(x) = 1$ if $\|x\| < 1/2, 0$ otherwise	Sinc: $f(\omega) = \operatorname{sinc}(\omega) = \sin(\pi\omega)/(\pi\omega)$
Gaussian: $f(x) = \mathrm{e}^{-\pi x^2}$	Gaussian: $f(\omega) = \mathrm{e}^{-\pi\omega^2}$
Constant: $f(x) = 1$	Delta: $f(\omega) = \delta(\omega)$
Sinusoid: $f(x) = \cos x$	Translated delta: $f(\omega) = \pi(\delta(1-2\pi\omega) + \delta(1+2\pi\omega))$
Shah: $f(x) = \text{Ш}_T(x) = T\sum_i \delta(x - Ti)$	Shah: $f(\omega) = \text{Ш}_{1/T}(\omega) = (1/T)\sum_i \delta(\omega - i/T)$

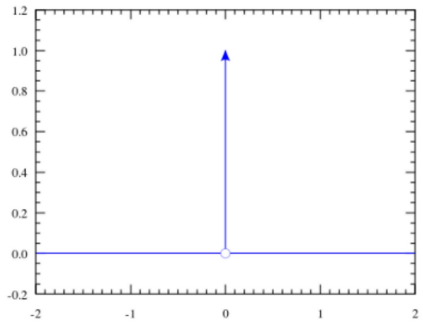
7.1.2 Ideal sampling and reconstruction

To obtain equally-spaced sample positions, a function is multiplied by an ***impulse train function or shah***:

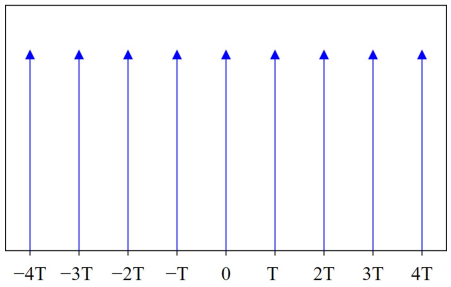
$$III_T(x) = T\sum_{k=-\infty}^{\infty}\delta(x - kT)$$

where T is the period or sampling rate, and δ is the Dirac delta distribution function.

A ***Dirac delta distribution function*** δ is defined such that $\int \delta(x)dx = 1$ and $\delta(x) = 0$ for all $x \neq 0$. Note that d is a distribution function, not a regular function (the graph of such a function would not have any area at all under the "curve", so it would be impossible for its definite integral to be 1; instead, it is thought of as the limit of a unit area box function centered at the origin with width approaching 0).

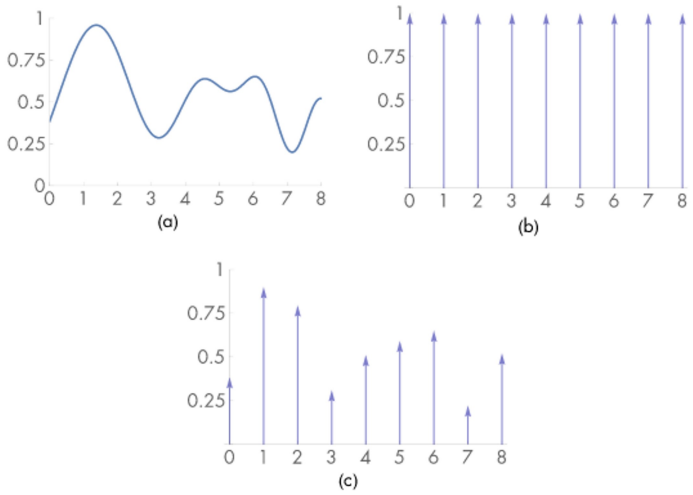


$\delta(x - kT)$, as in the definition of the shah function, is the delta function shifted left or right by a multiple of the period given by the index of summation k.



An infinite sequence of values of the function f result from multiplying it by the shah function:

$$III_T(x)f(x) = T\sum_k \delta(x - kT)f(kT)$$



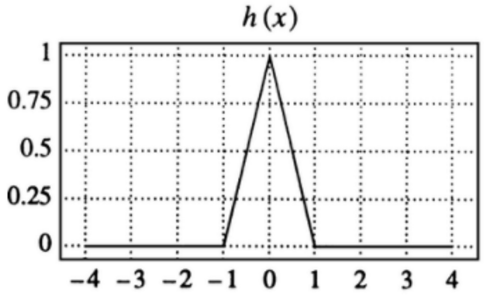
The reconstructed function $\tilde{f}$ is given by the **convolution** of this infinite sequence of (sample) values of f and a reconstruction filter r:

$$III_T(x)f(x) * r(x)$$

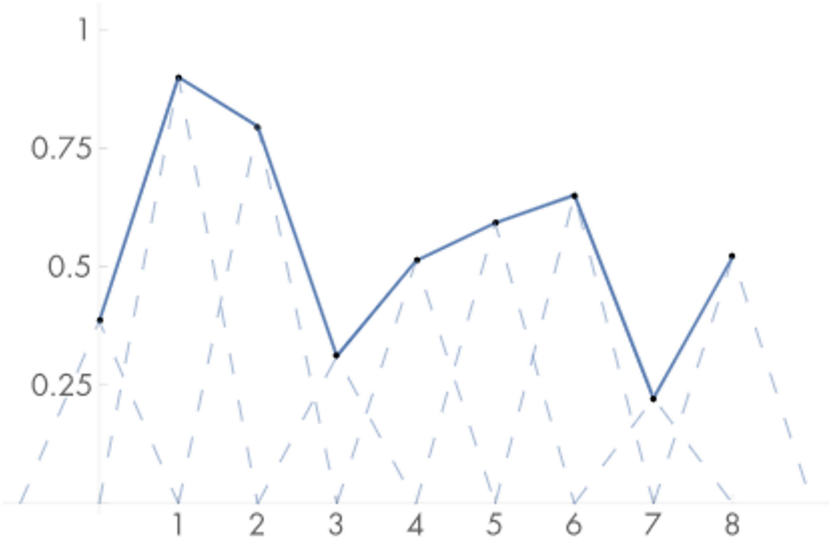
The convolution is equivalent to a weighted sum of scaled instances of the reconstruction filter centered at the sample points (the $r(x - kT)$ factor is the filter shifted to a sample point; the $f(kT)$ factor is the value of the sample):

$$\tilde{f}(x) = T\sum_{k=-\infty}^{\infty} f(kT)\, r(x - kT)$$

For example, $r(x) = \max(0, 1 - |x|)$ defines a triangle filter, whose graph is a triangle of height 1 and base 2 centered at the origin:



The convolution shifts the triangle to the sample point and scales it by the sample value:



Note that the same can be achieved by ***linear interpolation*** of the sample values.

A function is ***band-limited*** if there is a frequency ω_0 such that the function doesn't contain frequencies greater than ω_0. For frequencies $|\omega| > \omega_0$, band-limited functions have a value of 0 in the frequency domain: $F(\omega) = 0$.

Convolution in the spatial domain is equivalent to multiplication in the frequency domain, and convolution in the frequency domain is equivalent to multiplication in the spatial domain:

$$F\{f(x)g(x)\} = F(\omega) * G(\omega)$$

$$F\{f(x) * g(x)\} = F(\omega)G(\omega)$$

The Fourier transform maps the shah function with period T in the spatial domain to a shah function with period $1/T$ in the frequency domain.

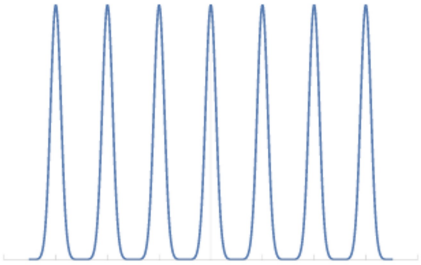
$$III_T(x) = T\sum_k \delta(x - kT) \;\rightarrow\; III_{1/T}(\omega) = \frac{1}{T}\sum_k \delta\left(\omega - \frac{k}{T}\right)$$

The consequence is that if the samples are farther apart in the spatial domain, they are closer together in the frequency domain.

Since multiplication in the spatial domain is convolution in the frequency domain, the product that gives the infinite sequence of samples is the convolution of the functions in frequency domain:

$$III_T(x)f(x) \rightarrow III_{1/T}(\omega) * F(\omega)$$

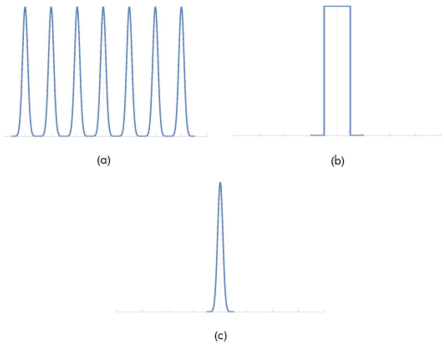
and the convolution is always an infinite series of copies of $F(\omega)$ (this is because convolving a function with a delta function yields a copy of the function):



Given the infinite series of copies of the frequency-domain $F(\omega)$, multiplication by the ***box function***

$$II_T(\omega) = \begin{cases} \frac{1}{T} & |\omega| < T/2 \\ 0 & otherwise \end{cases}$$

isolates the copy centered at the origin, that is, the original $F(\omega)$.



That's in the frequency domain. The inverse Fourier transform maps the box function $II_{1/T}(\omega)$ to the $sinc_T(x)$ function in the spatial domain.

Assuming that $f(x)$ is band-limited, its reconstruction in the frequency domain $\tilde{F}$ is computed as follows:

$$\tilde{F} = \left(III_{\frac{1}{T}}(\omega) * F(\omega)\right) II_{\frac{1}{T}}(\omega)$$

where, as shown before, the first factor is the infinite series of copies of the spectrum of f and the second factor is the box function that isolates the copy centered at the origin.

In the spatial domain, the inverse Fourier transform maps $\tilde{F}$ to $\tilde{f}$ (the reconstruction of f) as follows:

$$\tilde{f} = I\left(I_T(x)f(x)\right) sinc_T(x)$$

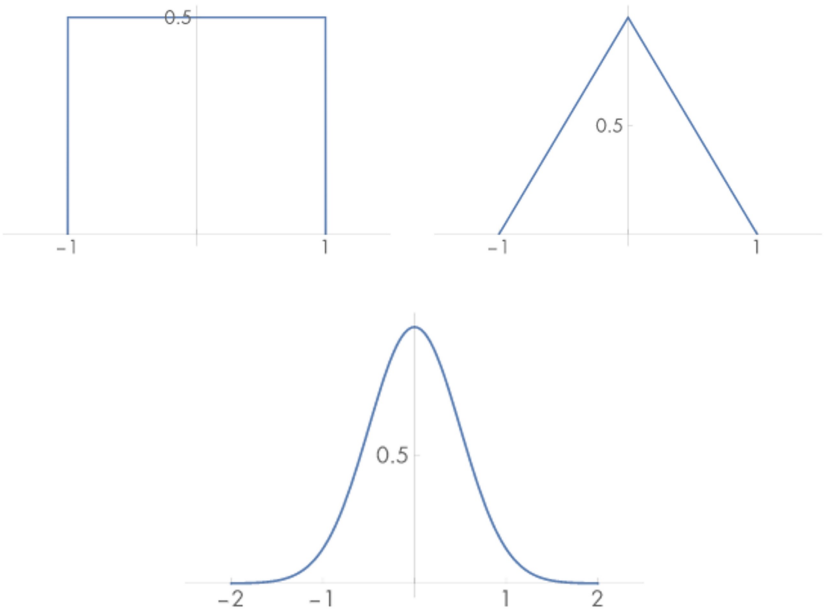
where $sinc_T(x)$ acts as the reconstruction filter in the spatial domain. It is shown above that this convolution is equivalent to a weighted sum of scaled instances of the reconstruction filter centered at the sample points:

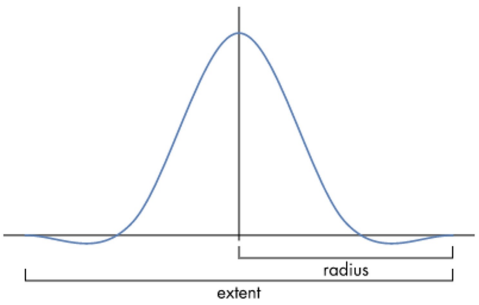
$\tilde{f} = \sum_{k=-\infty}^{\infty} f(kT)\ sinc(x - kT)$

This process describes the ***ideal way*** of doing sampling and reconstruction. Given that the sum uses an ***infinite number of samples*** (spaced equally by T) and that the function ***must be band-limited***, sampling and reconstruction ***can't be done that way*** in rendering.

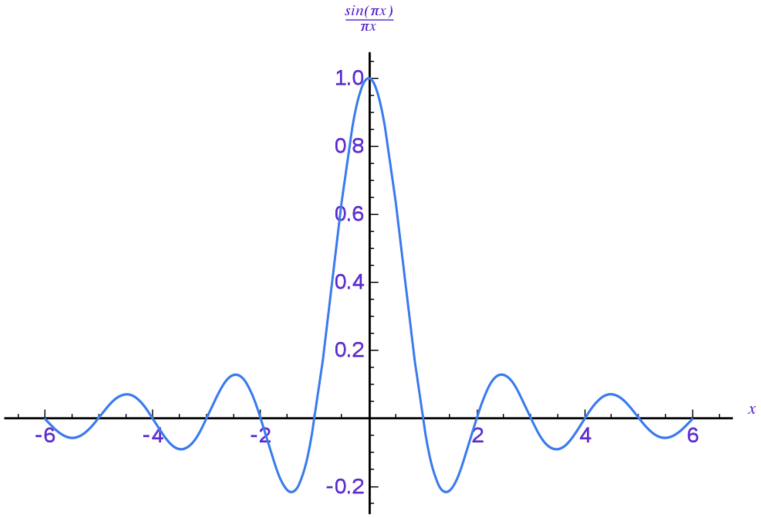
$sinc$ is said to have ***infinite extent or support***. The filter's extent is the interval that it spans in both directions before becoming or ***falling off*** to 0. For example, the extent of these box and triangle filters is $[-1,1]$ and the extent of this Gaussian filter is $[-2,2]$. On the other hand, the extent of $sinc$ is infinite:

Box, triangle, and Gaussian filters finite extents





sinc's infinite extent

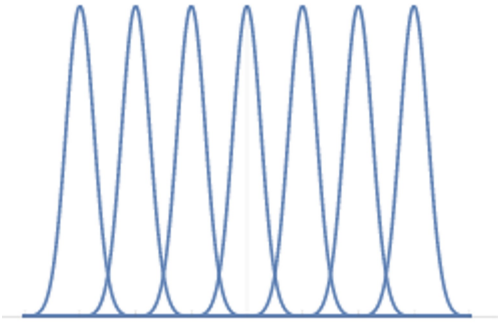


$sinc's$ infinite extent means that it needs all the uniformly spaced samples at once to perform the reconstruction, just as the summation formula of $\tilde{f}$ shows. That's another reason why it can't be used in rendering.

7.1.3 Aliasing

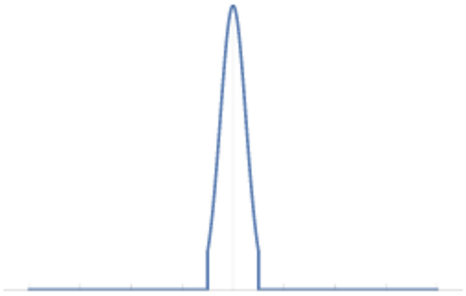
In signal processing, an ***alias*** is a signal that is reconstructed from samples of an original signal and that is not exactly the same as the original due to an improper sampling rate. In rendering, aliasing happens because both sampling and reconstruction are approximations; ***pre-aliasing*** is the result of bad sampling and ***post-aliasing*** is the result of bad reconstruction.

The key to successful reconstruction is recovering the original spectrum $F(\omega)$ out of the infinite series of copies, by multiplying the series by a box function of the appropriate width. This requires that the signal be sampled at the right sampling rate T. If the sampling rate T used by the spatial-domain shah function is too low (that is, the spacing between samples is too large; ***undersampling***), $1/T$ in the frequency-domain shah will be too high, and the copies of the spectrum $F(\omega)$ will be very close together and possibly overlapping:

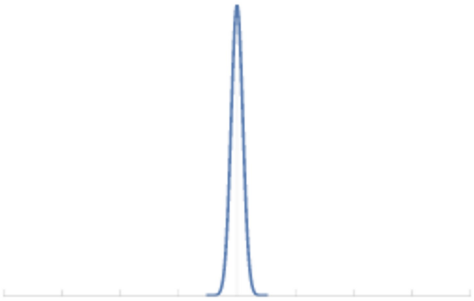


Even if the box function is of the right width, the isolated copy of the spectrum will be an ***alias***:

Alias



Correct spectrum



The ***Sampling Theorem*** tells us that perfect reconstruction of a ***band-limited*** signal from samples is possible if the sampling rate is at least twice the maximum frequency w_0 present in the signal. This minimum rate is called the ***Nyquist frequency***.

For ***non-band-limited*** signals, though, the Nyquist frequency does not exist (it's infinite: $2w_0 = \infty$). The image function, for example, has infinite-frequency components.

7.1.4 Antialiasing techniques

The obvious antialiasing technique for band-limited signals is increasing the sampling rate up to the Nyquist frequency.

For non-band-limited signals, some of the techniques are: nonuniform sampling, adaptive sampling, and prefiltering.

- *Nonuniform sampling*: the shah impulse train is modified to sample at an irregular, nonuniform rate:

  $$\sum_{k=-\infty}^{\infty} \delta(x - (k + \frac{1}{2} - \xi)T)$$

  where $(x - \left(k + \frac{1}{2}\right))$ is the midpoint of an otherwise regular interval and ξ is a random number in $[0,1]$ that displaces it.

  The result in frequency-domain is a series of copies of the spectrum $F(\omega)$ spaced nonuniformly and overlapping at random points. Visually, this is perceived as ***noise*** rather than coherent aliasing.

- *Adaptive sampling*: TODO

- *Prefiltering*: TODO

7.1.5 Application to image synthesis

We introduced the theory of sampling and reconstruction for 1D functions. In rendering, an image is a ***multidimensional*** function of radiance L:

$$f(x, y, t, u, v, i_1, i_2, \dots) \rightarrow L$$

where x and y are film plane or image pixel coordinates; t is time (relevant when the scene is in motion); u and v are lens coordinates; and i_k are various statistical quantities used by integrators and light transport algorithms.

The image function f is usually not band-limited. If sampled at a ***fixed, pixel-size rate***, the result will be a coherent alias. The antialiasing techniques that PBRT uses are ***supersampling*** and ***nonuniform sampling.***

Supersampling is done by sampling at the subpixel level.

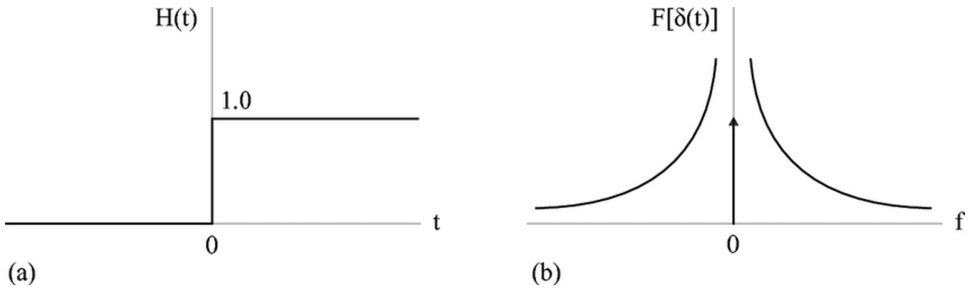
Nonuniform sampling results in noise, which is more pleasing to the human eye than coherent aliasing.

Good samples for a given image pixel are different from each other. We want each sample to capture different information about the image function.

7.1.6 Sources of aliasing in rendering

When geometry gets projected onto the image plane, it introduces a ***step function***: the image function's value instantaneously jumps from one value to another. This is known as ***geometric aliasing***.

Look at the frequency-domain representation of a step function centered at the origin: it has an infinite value at the corresponding frequency (**I'm actually not sure how to interpret the graphs?**).



Sharp shadows also introduce another step function.

Aliasing in rendering can't be eliminated, only mitigated.

7.1.7 Understanding pixels

Thinking about pixels as small squares with finite area is a ***dangerous mental model***.

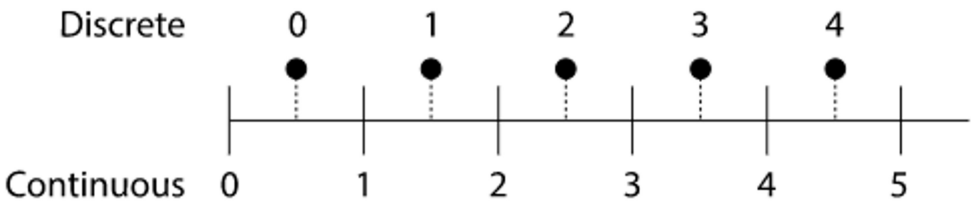
Image coordinates (x, y) may have discrete components (as when addressing a pixel of the film) or be continuous and have real-valued components (the coordinate of a sample, for example).

PBRT converts between continuous coordinates c and discrete coordinates d as follows:

$d = \lfloor c \rfloor$

$c = d + 1/2$

So the discrete range $[x_0, x_1]$ is mapped to the continuous range $[x_0, x_1 + 1)$.



7.2 Sampling interface

The Sampler class is the interface of all the sampler implementations.

The domain of the image function is the set of all n-dimensional vectors of the form $[0,1)$, the meaning of each dimension being $(x, y, t, u, v, i_1, i_2, \dots)$.

Samplers produce sequences of these $[0,1)$ n-dimensional ***sample vectors***.

Drawing numbers from $[0,1)$ uniformly at random is a sampling method; it is effective, but not efficient: many such samples are needed. With a carefully designed sampler, fewer samples are required to achieve a good image quality, and they are typically not much more expensive than uniform samplers.

7.2.1 Evaluating sample patterns: discrepancy

TODO: advanced section.

7.2.2 Basic sampler interface

In code.

The basic Sampler interface allows for the film to be decomposed into small ***image tiles*** and is ***well suited for multi-threaded rendering***, where each thread computes pixels in a local region that can be efficiently merged into the final image.

*Single samples (1 1D or 1 2D sample) or array-samples (*blocks of multiple $[0,1)$ samples*)* may be requested. The samples in array-samples are generated all at once, which gives the Sampler an opportunity to employ ***sample placement techniques***, which improve their distribution.

7.2.4 Pixel sampler

PixelSamplers generate samples for a single pixel at a time.

The PixelSampler class is the base class for all such sampling algorithms.

ZeroTwoSequenceSampler, StratifiedSampler, MaxMinDistSampler are all PixelSamplers.

7.2.5 Global sampler

GlobalSamplers generate consecutive samples that are spread across the entire image, visiting completely different pixels in succession. Without the GlobalSampler interface, it'd be very awkward to do this kind of sampling using the basic Sampler interface.

HaltonSampler and SobolSampler are GlobalSamplers.

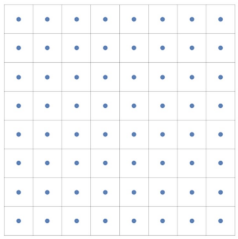
7.3 Stratified sampling

The StratifiedSampler subdivides a pixel area into nonoverlapping rectangular regions of equal size called ***strata*** and generates one and only one sample from each region, effectively capturing different and nonredundant information about the image function.

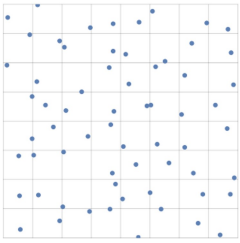
The smaller the strata, the higher the sampling rate is.

Stratified sampling may be uniform or ***nonuniform***: uniform if the sample point is the center of the stratum; nonuniform if the sample point is the ***jittered center*** of the stratum. Nonuniform stratified sampling is what's used in practice, because it ***turns aliasing into noise***.

Uniform stratified sampling

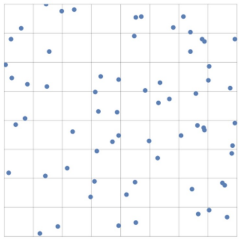


Jittered stratified sampling



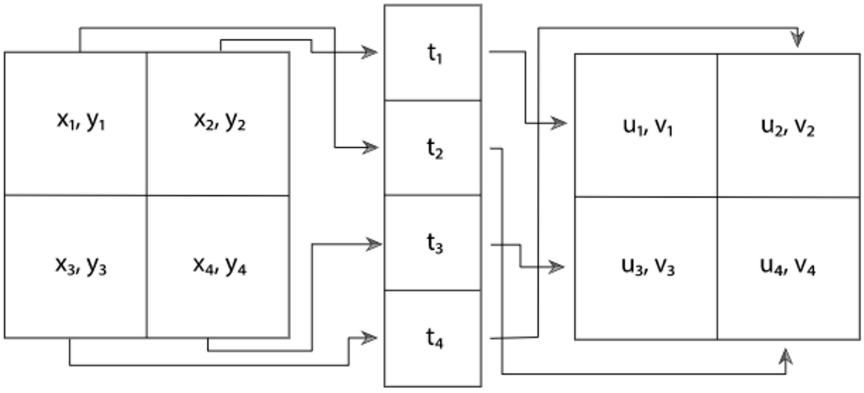
For reference, this image shows a stratified pixel, but the strata aren't used for sampling. Instead, sampling is done at random. Notice that some strata aren't sampled at all, some are undersampled, and some have clumps of samples. This is bad sampling.

Random sampling



Stratifying n dimensions with s strata results in $O(s^n)$ samples, which is too much **(but does every one of these samples have to be generated?).** This the ***curse of dimensionality***.

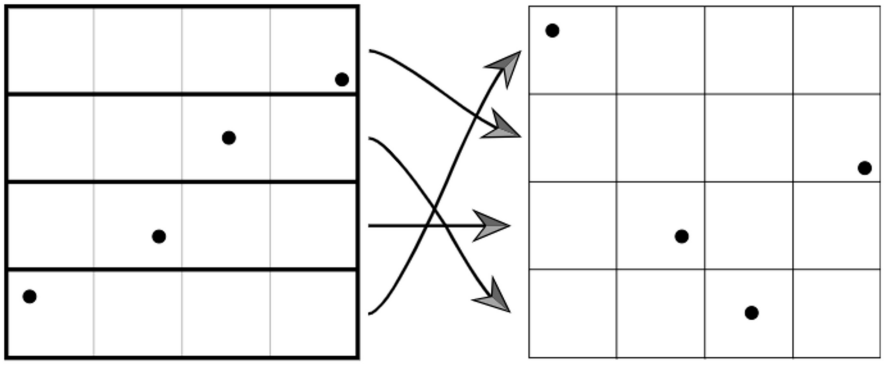
To avoid the curse of dimensionality, a good sample pattern can be generated by stratifying each dimension independently and associating samples of random strata of different dimensions. Here, for example, we have the 5 dimensions that a Camera sample uses: film point (x, y), time t, and lens point (u, v). A sample from the 3rd film point stratum is associated at random with a sample from the 4th time stratum and a sample from the 2nd lens point stratum. This is called ***padding***. Instead of a 5D stratification, we have 5 dimensions stratified independently.



Array-samples may be requested with an arbitrary size. These pose a problem for stratification when the size n of the array-sample is not the number of cells of a possibly multi-dimensional grid (that is, a square of an integer). The only stratification patterns possible are $n\ x\ 1$ and $1\ x\ n$, which are 1-dimensional.

The Sampler interface offers RoundSize(), which gives callers the opportunity to request array-samples of a size adequate for stratification. But StratifiedSampler chooses to support arbitrarily-sized array-samples, using ***Latin hypercube sampling (LHS***), also known as ***n-rooks sampling***.

LHS chooses samples such that they are the only ones on their column and row. This can be done by generating random samples in the cells along the diagonal and then randomly permuting their coordinates.



The book discusses the limitations of LHS.

7.8 Image reconstruction

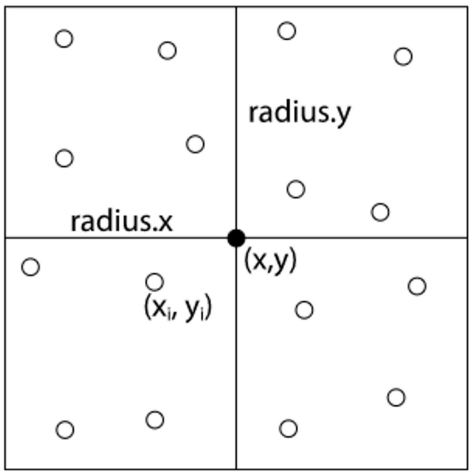
Applying ***perfect reconstruction*** (i.e. using the $sinc$ function, etc) techniques to image samples for image

synthesis does not result in the highest quality because the image function is not band-limited.

Image reconstruction is the process of taking the samples taken for a pixel and computing a final value for it. This process is an ***interpolation*** of the samples' radiance values.

The sampling space of a given pixel generally overlaps with that of the neighboring pixels. The sample points that participate in the interpolation exist in an area around the pixel's location defined by 2 radiuses, one in the x direction and one in the y direction. In signal processing theory, this area defined by the 2 radiuses is the ***filter's extent or support***.

Interpolation is implemented with a ***weighted average***, across all the samples of the filter's extent, of the weighted product of the radiance function $L(x_i, y_i)$ evaluated at the sample point (x_i, y_i) and the filter $f(x - x_i, y - y_i)$, where (x, y) is the pixel's location as shown here:



Reconstructed pixel value via weighted average interpolation:

$$I(x, y) = \frac{\sum_i f(x - x_i, y - y_i)\, w(x_i, y_i)\, L(x_i, y_i)}{\sum_i f(x - x_i, y - y_i)}$$

7.8.1 Filter functions

The filters here are defined in the spatial domain, not in the frequency domain. That is, their input are points in space, not frequencies.

It must be noted that input sample points have coordinates with respect to the filter's center and ***normalized to lie within*** the filter's extent.

Box filter

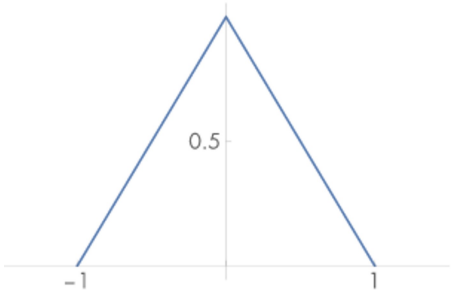
A box filter maps sample points to a constant value. The reconstruction weighted average is thus not weighted at all.

The box filter does a very poor reconstruction job.

Triangle filter

A triangle filter maps sample points a value that decreases linearly as the distance of the sample point to the filter's center increases.

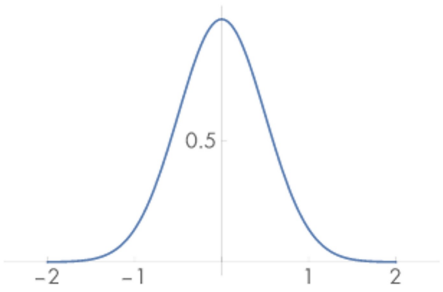
A sample point just at the center, $(x_i, y_i) = (0,0)$ gets mapped to $r_x * r_y$, the maximum value, where r is the filter's radius. All other sample points get mapped to a smaller value.



Gaussian filter

A Gaussian filter applies a ***Gaussian bump*** that is centered at the pixel and is radially symmetric around it.

The Gaussian filter maps sample points to values of a bell curve that decrease nonlinearly as the coordinates of the sample point tend to the filter's radius, at which point the function's value is 0.



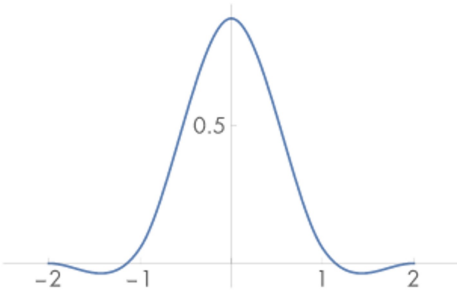
$f(x) = e^{-\alpha x^2} - e^{-\alpha r^2}$

The first term of the sum is the Gaussian function evaluated at the sample's coordinate. The second term is the Gaussian function evaluated at the radius. The subtraction is the filter's value for the coordinate and ensures that it falls off to 0 as the coordinate approaches the radius.

The Gaussian filter produces blurrier images than other filters, but most of the time the blur masks the aliasing that remains after the reconstruction.

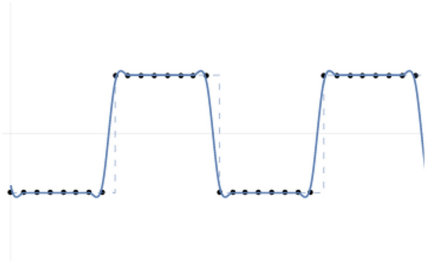
Mitchell -Netravali filter

A Mitchell-Netravali filter does a good job of trading off between ***ringing*** (phantom edges next to actual edges) and ***blurring***.

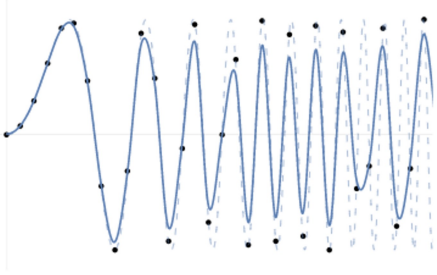


The Mitchell-Netravali filter has a bell-shaped curve that, unlike the Gaussian, has ***negative lobes***. The negative lobes give sharpness to edges.

Very good reconstruction of a step function



Good reconstruction of a sinusoidal, except when the frequency became higher than the sampling rate



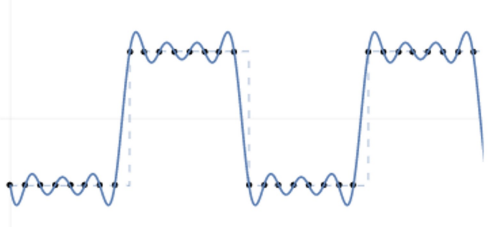
The Mitchell-Netravali function is piecewise defined over 4 subdomains: $[-2,-1], [-1,0], [0,1], [1,2]$. And parameterized by B and C. The values of B and C are chosen carefully to avoid jump discontinuities at the boundaries of the subdomains. They are chosen to satisfy $B + 2C = 1$ (is the function continuous for any values of B and C that satisfy this equation?).

$$f(x) = \frac{1}{6}\begin{cases} (12 - 9B - 6C)|x|^3 + (-18 + 12B + 6C)|x|^2 + (6 - 2B) & |x| < 1 \\ (-B - 6C)|x|^3 + (6B + 30C)|x|^2 + (-12B - 48C)|x| + (8B + 24C) & 1 \le |x| < 2 \\ 0 & \text{otherwise.} \end{cases}$$

Windowed sinc filter

The $sinc$ filter used by the perfect sampling and reconstruction method can't be used in rendering because: 1) it has infinite extent, and 2) it introduces ***ringing*** when used with a finite number of samples.

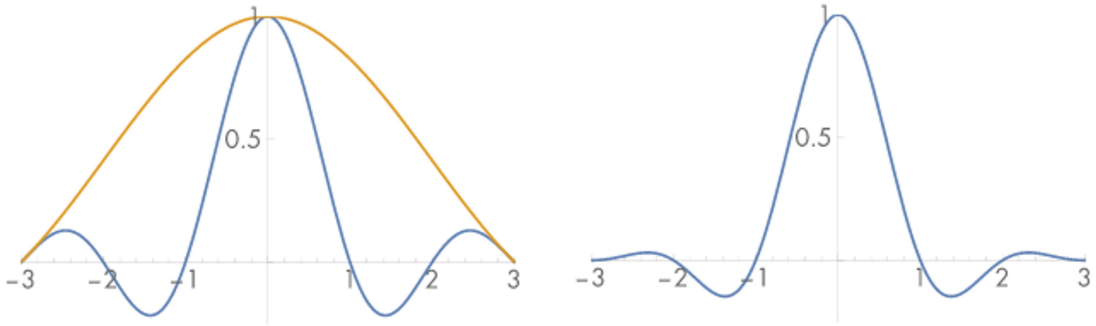
Ringing caused by undersampling



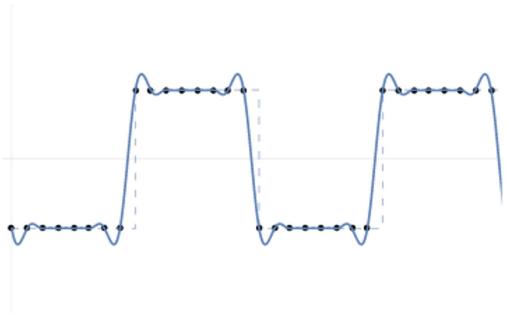
The windowed $sinc$ filter uses the ***Lanczos windowing function*** to ***limit the extent*** of the $sinc$ filter to a

given number of cycles τ and to make it ***fall off*** to 0. The Lanczos function is also a $sinc$ function that has been scaled in a way such that its central lobe spans τ cycles of the filter's $sinc$ function:

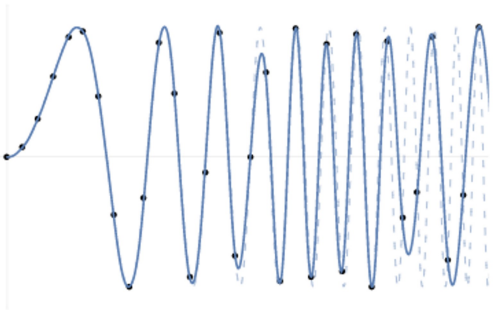
Note that the product (right) of the sinc and the Lanczos windowing function (left) has different shape than the original sinc



The windowed sinc filter reconstructs a step function with much less ringing



Good reconstruction of a sinusoidal, except where undersampled



7.9 Film and the imaging pipeline

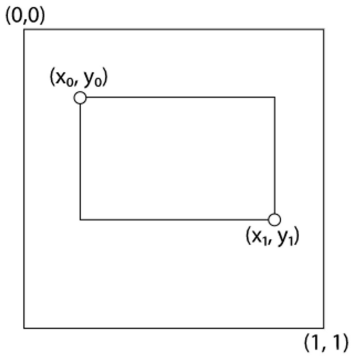
The Film (sensor) class maintains a representation of the image, updating it with each ray's sample.

When the image is complete, the main rendering loop exits and the Film class writes the image to a file. The image stores floating-point values (the SPD coefficients?), because the traditional 8-bit unsigned integer image format loses information.

For nonrealistic cameras, the sensor ***measures*** the average radiance over the sensor area over a period of time. For realistic ones, it measures the exact amount of energy that arrives over the sensor area over a period of time. These differences are encapsulated by the Camera implementations (what are the differences exactly?).

The final value of the pixel is computed by the Film using the reconstructed pixel value equation.

The ***crop window*** is the subset of the full image resolution that is actually rendered (usually the crop window is the full image resolution). It is specified as a bounding box in NDC space coordinates.



Pixel structures store the result of the reconstruction of the image function from samples. The result is a color. It could have been a SampledSpectrum or an RGBSpectrum, but the choice was to store only the 3 XYZ tristimulus values. The authors say that there's no loss of information. They chose to store XYZ over RGB tristimulus values to be device-independent, as opposed to assuming a certain display's RGB response curves. Only when saving to a file will RGB be used.

INTERESTING: A sample may be within the radius of the filters of at least 16 pixels and contribute to the color of all of them.

IMPORTANT: PBRT does not evaluate the reconstruction filter at every sample to obtain the sample's filter weight at rendering time. Instead, it precomputes it for $16 \, x \, 16$ uniformly distributed sample points over the 2D extent of the filter at Film construction time. And then at rendering time it looks up the filter weight of the precomputed sample point that is closest to the actual sample point.

The sampling area extends beyond the region of the scene captured by the film: pixels at or near the boundaries of the film or the crop window need those samples to be reconstructed properly because they are in the extent of the filter. In the case of images rendered in tiles using crop windows, aliasing appears at the joints if those samples aren't used.

7.9.2 Supplying pixel values to the film

Film tiles allow multiple threads to update the film concurrently with sample contributions. The FilmTile object stores the sums of spectral samples and corresponding filter weights generated for a region of the image by a single thread. When the thread is done, it passes it to the Film for merging.

Each pixel of a FilmTile keeps the running sums that appear in the numerator and denominator of the pixel reconstruction/filtering equation.

Rendering threads AddSample's to the FilmTile. Each sample makes a contribution to a number of pixels in the tile, weighted by the filter (in the cases of GaussianFilter, MitchellFilter and TriangleFilter, the filter weight decreases the farther away the sample is from the center of the filter, i.e. the center of the discrete pixel). The larger the extent of the filter, the more pixels a sample contributes to.

When the thread is done sampling and filtering a tile, the Film merges the tile with the entire image. Boundary pixels and pixels near the boundary of a tile may be within the extent of the filter of pixels on another tile. So rather than just copying the tile pixels (FilmTilePixel) over to the image pixels (Pixel), the Film sums the contributions of the tile to these pixels on top of the contributions made by other tiles.

- [ ] TODO: what's splatting?

7.9.3 Image output

The most important thing that's done when creating an image file is the conversion from XYZ colors to RGB, because there aren't many image file formats for XYZ.

Recall that XYZ tristimulus values are device-independent and RGB depends on the device's RGB response curves. Since it looks like it isn't possible to query the system for a display's RGB response curves, PBRT just uses the ***sRGB response curves***; sRGB is supposed to be supported by virtually all + 2015 displays and printers.

The conversion from XYZ to RGB is a change of spectral basis, where the basis functions are the device's RGB response curves.