2D Computer Graphics

Diego Nehab

Summer 2020

IMPA

Anti-aliasing and texture mapping

Anti-aliasing

Let f be a function and ψ an *anti-aliasing filter*

Anti-aliasing

Let f be a function and ψ an *anti-aliasing filter*

Value of pixel p_i is given by

$$p_i = (f * \psi)(i) = \int_{-\infty}^{\infty} f(t)\,\psi(i-t)\,dt$$

Let f be a function and ψ an *anti-aliasing filter*

Value of pixel p_i is given by

$$p_i = (f * \psi)(i) = \int_{-\infty}^{\infty} f(t)\,\psi(i-t)\,dt$$

How to compute the integral when f is a vector graphics illustration?

Analytic antialiasing

Assume **box** filter, single layer, solid color, simple polygon

- Clip polygon against the **box** centered at each pixel
- Compute weighted area using on Green's theorem from Calculus

Assume **box** filter, single layer, solid color, simple polygon

- Clip polygon against the **box** centered at each pixel
- Compute weighted area using on Green's theorem from Calculus

Possible to clip edges, not the shapes

- + general piecewise polynomial filters [Duff, 1989]
- + curved edges [Manson and Schaefer, 2013]

Assume **box** filter, single layer, solid color, simple polygon

- Clip polygon against the **box** centered at each pixel
- Compute weighted area using on Green's theorem from Calculus

Possible to clip edges, not the shapes

- + general piecewise polynomial filters [Duff, 1989]
- + curved edges [Manson and Schaefer, 2013]

What about polygons with self-intersections?

Analytic antialiasing

Assume **box** filter, single layer, solid color, simple polygon

- Clip polygon against the **box** centered at each pixel
- Compute weighted area using on Green's theorem from Calculus

Possible to clip edges, not the shapes

- + general piecewise polynomial filters [Duff, 1989]
- + curved edges [Manson and Schaefer, 2013]

What about polygons with self-intersections?

What about spatially varying colors?

Assume **box** filter, single layer, solid color, simple polygon

- Clip polygon against the **box** centered at each pixel
- Compute weighted area using on Green's theorem from Calculus

Possible to clip edges, not the shapes

- + general piecewise polynomial filters [Duff, 1989]
- + curved edges [Manson and Schaefer, 2013]

What about polygons with self-intersections?

What about spatially varying colors?

What about multiple opaque layers?

Analytic antialiasing

Assume **box** filter, single layer, solid color, simple polygon

- Clip polygon against the **box** centered at each pixel
- Compute weighted area using on Green's theorem from Calculus

Possible to clip edges, not the shapes

- + general piecewise polynomial filters [Duff, 1989]
- + curved edges [Manson and Schaefer, 2013]

What about polygons with self-intersections?

What about spatially varying colors?

What about multiple opaque layers?

What about transparency?

Popular hack

Assume path P_i with constant color f_i, α_{f_i}

Popular hack

Assume path P_i with constant color f_i, α_{f_i}

Assume blending over the background b_i, α_{b_i}

Popular hack

Assume path P_i with constant color f_i, α_{f_i}

Assume blending over the background b_i, α_{b_i}

Assume anti-aliasing filter ψ with support Ω

Popular hack

Assume path P_i with constant color f_i, α_{f_i}

Assume blending over the background b_i, α_{b_i}

Assume anti-aliasing filter ψ with support Ω

Define the *coverage* o of P_i at pixel p

$$o = \int_\Omega [u - p \in P_i] \, \psi(u) \, du$$

Assume path P_i with constant color f_i, α_{f_i}

Assume blending over the background b_i, α_{b_i}

Assume anti-aliasing filter ψ with support Ω

Define the *coverage* o of P_i at pixel p

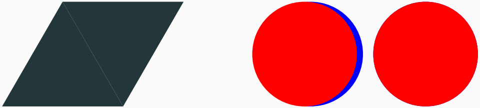
$$o = \int_\Omega [u - p \in P_i]\, \psi(u)\, du$$

The new background b_{i+1}, α_{i+1} is

$$b_{i+1}, \alpha_{i+1} = f_i, (\alpha_i \cdot o) \oplus b_i, \alpha_i$$

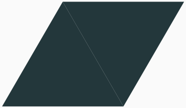
Problems with hack

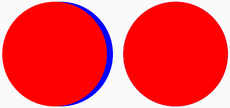
Visible seams at perfectly abutting layers, weird halos



This is called the *correlated mattes* problem

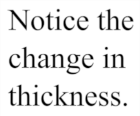
Visible seams at perfectly abutting layers, weird halos

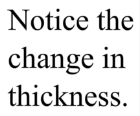




This is called the *correlated mattes* problem

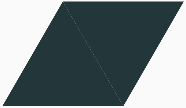
It also either blends in linear, or antialiases in gamma

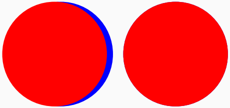




Problems with hack

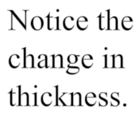
Visible seams at perfectly abutting layers, weird halos

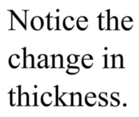




This is called the *correlated mattes* problem

It also either blends in linear, or antialiases in gamma





Must blend in gamma and antialias in linear [Nehab and Hoppe, 2008]

$$b_{i+1}, \beta i + 1 = \gamma\left(\gamma^{-1}(f_i, \alpha_i \oplus b_i, \beta_i) \cdot o + \gamma^{-1}(b_i, \beta_i) \cdot (1 - o)\right)$$

A *random variable* X is a function that maps outcomes to numbers

A *random variable* X is a function that maps outcomes to numbers

The associated *cumulative distribution function* F_X is such that

$$F_X(a) = P[X \leq a]$$

Probability in 2 slides

A *random variable* X is a function that maps outcomes to numbers

The associated *cumulative distribution function* F_X is such that

$$F_X(a) = P[X \leq a]$$

i.e., it measures the probability that the numerical value is at most a.

Probability in 2 slides

A *random variable* X is a function that maps outcomes to numbers

The associated *cumulative distribution function* F_X is such that

$$F_X(a) = P[X \leq a]$$

i.e., it measures the probability that the numerical value is at most a.

The associated *probability density function* f_X is such that

$$F_X(a) = \int_{-\infty}^{a} f_X(t)\, dt$$

A *random variable* X is a function that maps outcomes to numbers

The associated *cumulative distribution function* F_X is such that

$$F_X(a) = P[X \leq a]$$

i.e., it measures the probability that the numerical value is at most a.

The associated *probability density function* f_X is such that

$$F_X(a) = \int_{-\infty}^{a} f_X(t)\, dt$$

i.e., its integral is the cumulative distribution function.

A *random variable* X is a function that maps outcomes to numbers

The associated *cumulative distribution function* F_X is such that

$$F_X(a) = P[X \leq a]$$

i.e., it measures the probability that the numerical value is at most a.

The associated *probability density function* f_X is such that

$$F_X(a) = \int_{-\infty}^{a} f_X(t)\, dt$$

i.e., its integral is the cumulative distribution function.

The associated expectation $E[X]$ (or mean μ_X) is

$$E[X] = \int_{-\infty}^{\infty} t\, f_X(t)\, dt = \mu_X \tag{1}$$

Probability in 2 slides

A *random variable* X is a function that maps outcomes to numbers

The associated *cumulative distribution function* F_X is such that

$$F_X(a) = P[X \leq a]$$

i.e., it measures the probability that the numerical value is at most a.

The associated *probability density function* f_X is such that

$$F_X(a) = \int_{-\infty}^{a} f_X(t)\, dt$$

i.e., its integral is the cumulative distribution function.

The associated expectation $E[X]$ (or mean μ_X) is

$$E[X] = \int_{-\infty}^{\infty} t\, f_X(t)\, dt = \mu_X \qquad (1)$$

i.e., the mean value weighted by the probability density function.

The associated variance $\mathbf{var}(X) = \sigma_X^2$ is

$$\mathbf{var}(X) = E[(X - \mu_X)^2] = E[X^2] - E^2[X] = \sigma_X^2$$

and the standard deviation is σ_X.

The associated variance $\textbf{var}(X) = \sigma_X^2$ is

$$\textbf{var}(X) = E[(X - \mu_X)^2] = E[X^2] - E^2[X] = \sigma_X^2$$

and the standard deviation is σ_X.

Measure how much the random variable deviates from the mean

The associated variance $\textbf{var}(X) = \sigma_X^2$ is

$$\textbf{var}(X) = E[(X - \mu_X)^2] = E[X^2] - E^2[X] = \sigma_X^2$$

and the standard deviation is σ_X.

Measure how much the random variable deviates from the mean

The sample average is $\overline{X}_n = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)$

The associated variance $\mathbf{var}(X) = \sigma_X^2$ is

$$\mathbf{var}(X) = E[(X - \mu_X)^2] = E[X^2] - E^2[X] = \sigma_X^2$$

and the standard deviation is σ_X.

Measure how much the random variable deviates from the mean

The sample average is $\overline{X}_n = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)$

Law of large numbers

$$\overline{X}_n \to \mu_X \quad \text{for} \quad n \to \infty$$

The associated variance $\mathrm{var}(X) = \sigma_X^2$ is

$$\mathrm{var}(X) = E[(X - \mu_X)^2] = E[X^2] - E^2[X] = \sigma_X^2$$

and the standard deviation is σ_X.

Measure how much the random variable deviates from the mean

The sample average is $\overline{X}_n = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)$

Law of large numbers

$$\overline{X}_n \to \mu_X \quad \text{for} \quad n \to \infty$$

Variance of sample average

$$\mathrm{var}(\overline{X}_n) = \mathrm{var}\left(\tfrac{1}{n}\sum X_i\right) = \tfrac{1}{n^2}\sum \mathrm{var}(X_i) = \frac{\sigma_X^2}{n}$$

Monte Carlo integration

Start by expressing an integral as the expectation of a random variable

Estimate expectation by sample mean

Monte Carlo integration

Start by expressing an integral as the expectation of a random variable

Estimate expectation by sample mean

Rely on law of large numbers

Monte Carlo integration

Start by expressing an integral as the expectation of a random variable

Estimate expectation by sample mean

Rely on law of large numbers

Let X be such that support of f_X is Ω

$$\int_\Omega g(t)\,dt = \int_\Omega \frac{g(t)}{f_X(t)} f_X(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \frac{g(X_i)}{f_X(X_i)}$$

Monte Carlo integration

Start by expressing an integral as the expectation of a random variable

Estimate expectation by sample mean

Rely on law of large numbers

Let X be such that support of f_X is Ω

$$\int_\Omega g(t)\,dt = \int_\Omega \frac{g(t)}{f_X(t)} f_X(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \frac{g(X_i)}{f_X(X_i)}$$

This is the basis of *supersampling*

The solution to our anti-aliasing problems

Supersampling

Let $g : \mathbb{R}^2 \to RGB$ map positions to linear color

Consider an *anti-aliasing kernel* ψ

Supersampling

Let $g : \mathsf{R}^2 \to RGB$ map positions to linear color

Consider an *anti-aliasing kernel* ψ

The *linear* color at pixel p is

$$c(p) = \int_\Omega g(p - q)\,\psi(q)\,dq$$

Let $g : \mathsf{R}^2 \to RGB$ map positions to linear color

Consider an *anti-aliasing kernel* ψ

The *linear* color at pixel p is

$$\begin{aligned} c(p) &= \int_\Omega g(p-q)\,\psi(q)\,dq \\ &= E\left[\frac{g(p-X)\psi(X)}{f_X(X)}\right] \end{aligned}$$

Let $g : \mathsf{R}^2 \to RGB$ map positions to linear color

Consider an *anti-aliasing kernel* ψ

The *linear* color at pixel p is

$$\begin{aligned} c(p) &= \int_\Omega g(p-q)\,\psi(q)\,dq \\ &= E\left[\frac{g(p-X)\psi(X)}{f_X(X)}\right] \\ &\approx \frac{1}{n}\sum_{i=1}^{n}\frac{g(p-X_i)\,\psi(X_i)}{f_X(X_i)} \end{aligned}$$

Let $g : \mathbb{R}^2 \to RGB$ map positions to linear color

Consider an *anti-aliasing kernel* ψ

The *linear* color at pixel p is

$$\begin{aligned} c(p) &= \int_\Omega g(p-q)\,\psi(q)\,dq \\ &= E\left[\frac{g(p-X)\psi(X)}{f_X(X)}\right] \\ &\approx \frac{1}{n}\sum_{i=1}^{n}\frac{g(p-X_i)\,\psi(X_i)}{f_X(X_i)} \end{aligned}$$

When $\psi = \beta^0$ is the box, $f_X = 1$ with support $\Omega = [-\frac{1}{2}, \frac{1}{2}]^2$

$$c(p) \approx \frac{1}{n}\sum_{i=1}^{n} g(p - X_i)$$

Biased estimator

Estimator is *unbiased* if expected value is correct

Biased estimator

Estimator is *unbiased* if expected value is correct

The Monte Carlo estimator is unbiased in this sense

$$c(p) \approx \frac{1}{n} \sum_{i=1}^{n} \frac{g(p - X_i)\, \psi(X_i)}{f_X(X_i)}$$

Biased estimator

Estimator is *unbiased* if expected value is correct

The Monte Carlo estimator is unbiased in this sense

$$c(p) \approx \frac{1}{n} \sum_{i=1}^{n} \frac{g(p - X_i)\, \psi(X_i)}{f_X(X_i)}$$

It often makes sense to use a *biased* estimator to reduce *variance*

$$c(p) \approx \frac{\displaystyle\sum_{i=1}^{n} \frac{g(p - X_i)\, \psi(X_i)}{f_X(X_i)}}{\displaystyle\sum_{i=1}^{n} \frac{\psi(X_i)}{f_X(X_i)}}$$

What happens if we choose $f_X(t) \propto g(t)$?

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right]$$

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha]$$

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha] = \frac{g(X)}{f(X)}$$

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha] = \frac{g(X)}{f(X)}$$

We only need one sample!

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha] = \frac{g(X)}{f(X)}$$

We only need one sample!

Unfortunately, we need to normalize g to transform it into a PDF

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha] = \frac{g(X)}{f(X)}$$

We only need one sample!

Unfortunately, we need to normalize g to transform it into a PDF

For that, we need to divide it by its integral

This integral is exactly what we are trying to compute!

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha] = \frac{g(X)}{f(X)}$$

We only need one sample!

Unfortunately, we need to normalize g to transform it into a PDF

For that, we need to divide it by its integral

This integral is exactly what we are trying to compute!

However, we can often make f_X *almost* proportional to g

Importance sampling

What happens if we choose $f_X(t) \propto g(t)$?

$$\int_\Omega g(t)\,dt = E\left[\frac{g(X)}{f_X(X)}\right] = E[\alpha] = \frac{g(X)}{f(X)}$$

We only need one sample!

Unfortunately, we need to normalize g to transform it into a PDF

For that, we need to divide it by its integral

This integral is exactly what we are trying to compute!

However, we can often make f_X *almost* proportional to g

This is *importance sampling*

Many different point distributions have $f_X = 1/A_\Omega$ in Ω

Many different point distributions have $f_X = 1/A_\Omega$ in Ω

Uniform, stratified, low-discrepancy (e.g. Poisson disk, Lloyd relaxation)

Better sample distributions

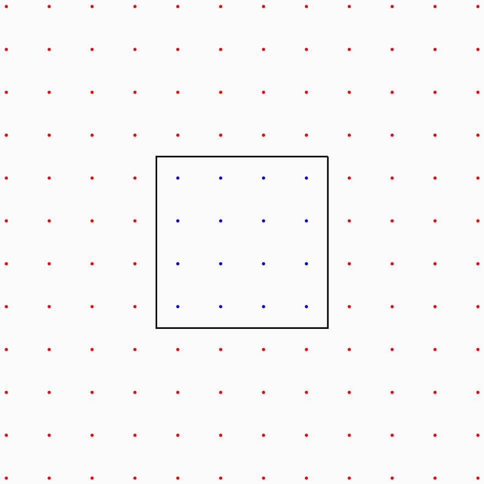
Many different point distributions have $f_X = 1/A_\Omega$ in Ω

Uniform, stratified, low-discrepancy (e.g. Poisson disk, Lloyd relaxation)

Variance of $\overline{X}_n$ is not the same for all of them!

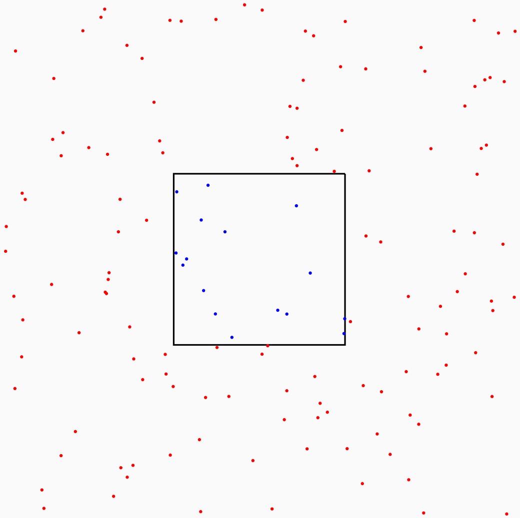
16 SAMPLES

Regular



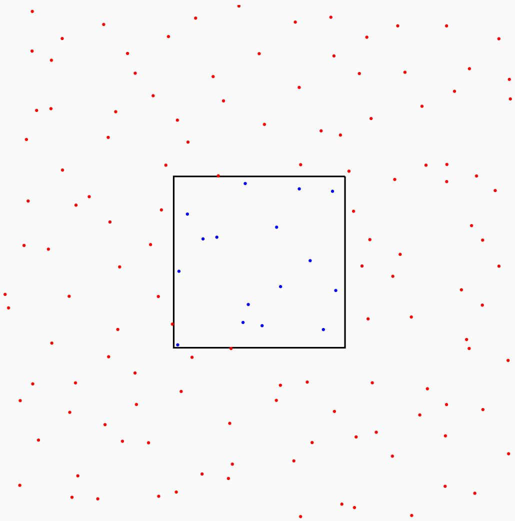
16 SAMPLES

Uniform



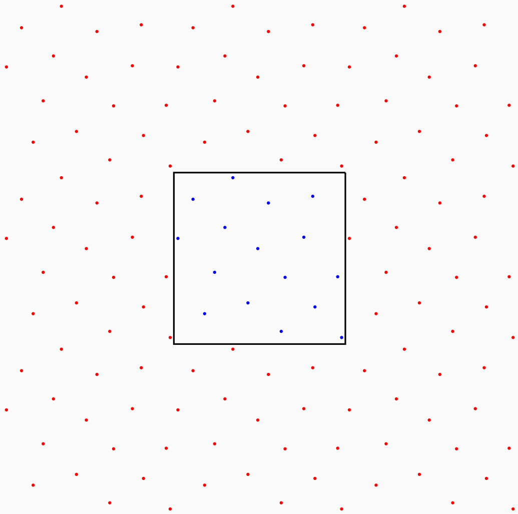
16 SAMPLES

Stratified



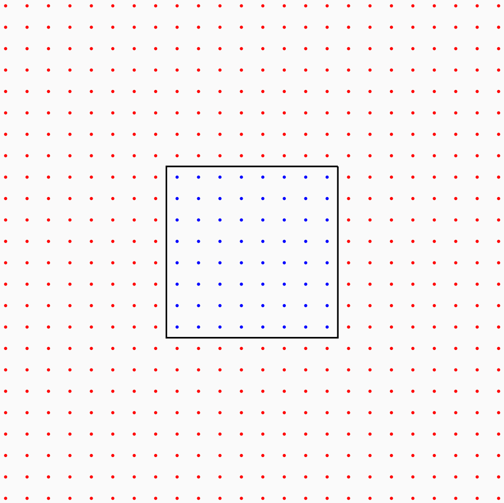
16 SAMPLES

Blue noise

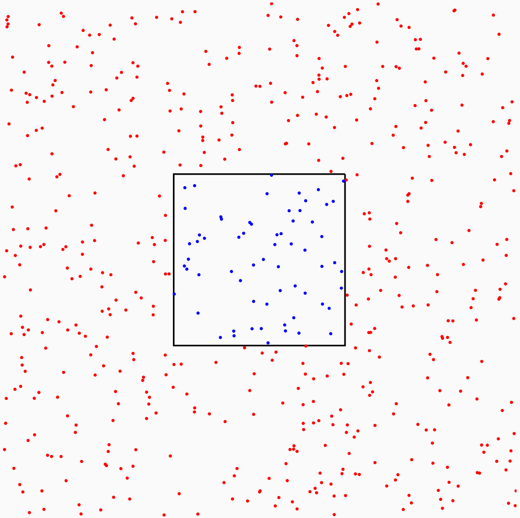


64 SAMPLES

Regular

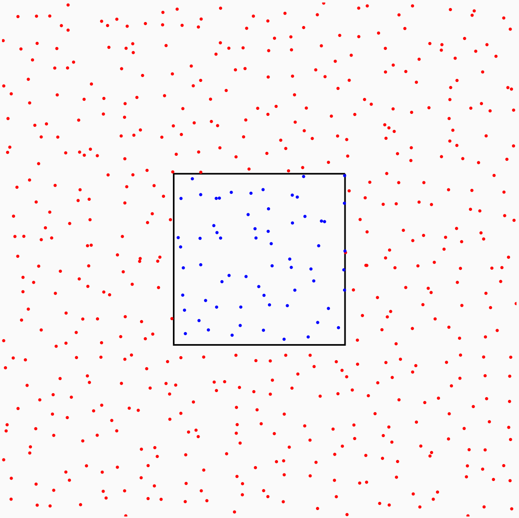


Uniform



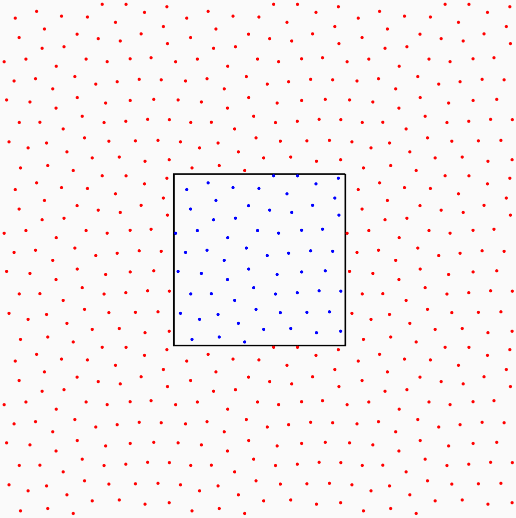
64 SAMPLES

Stratified



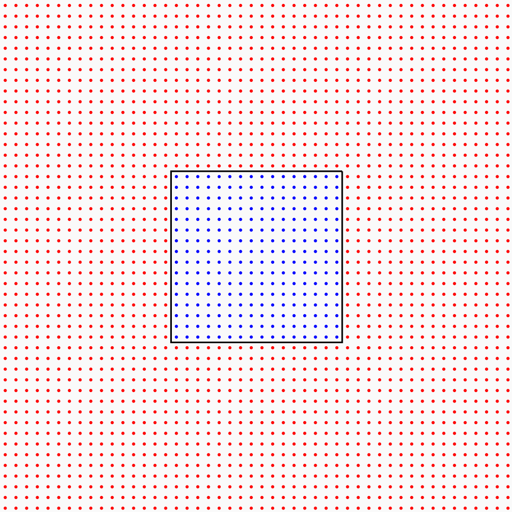
64 SAMPLES

Blue noise



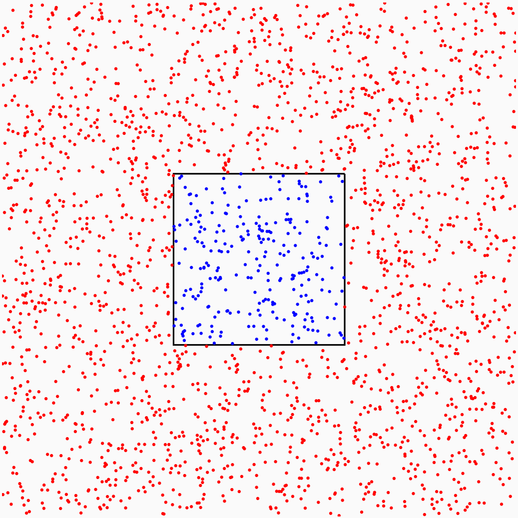
256 SAMPLES

Regular



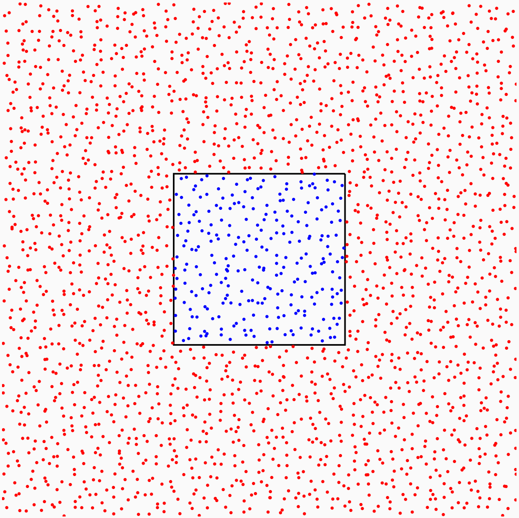
256 SAMPLES

Uniform

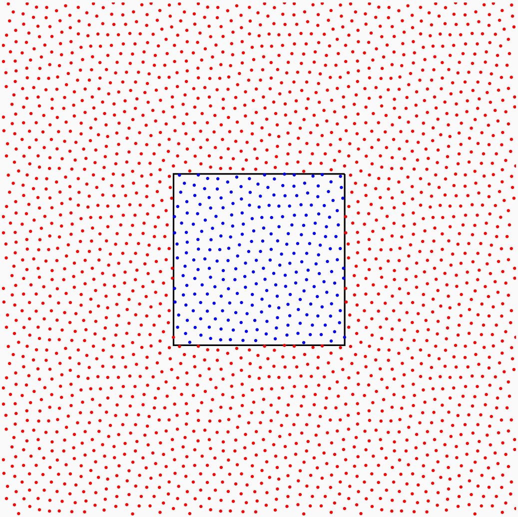


256 SAMPLES

Stratified

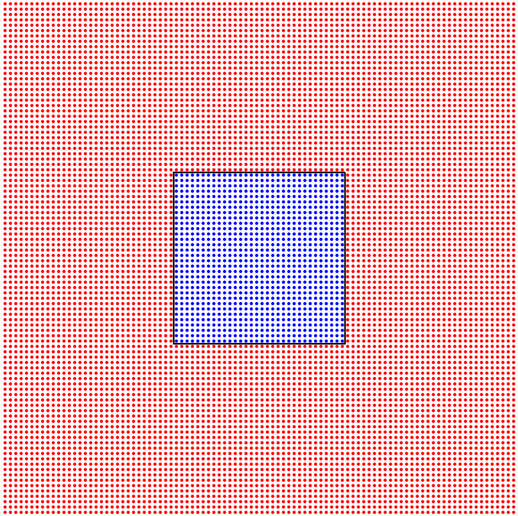


Blue noise

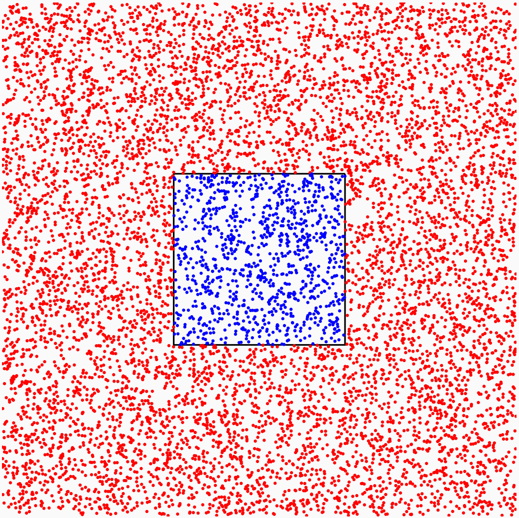


1024 samples

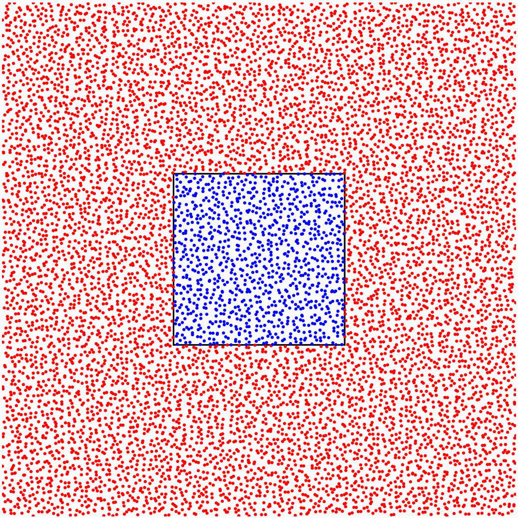
Regular



Uniform

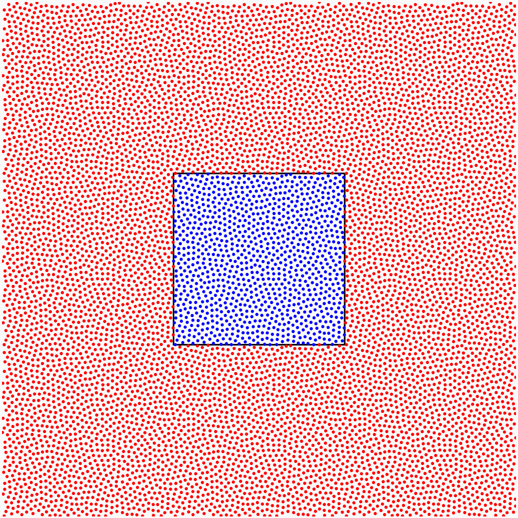


Stratified

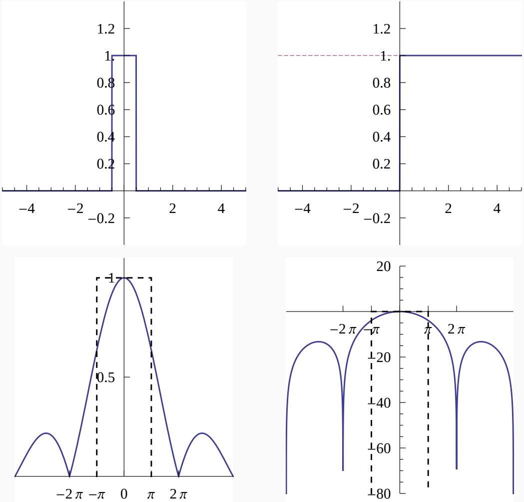


1024 SAMPLES

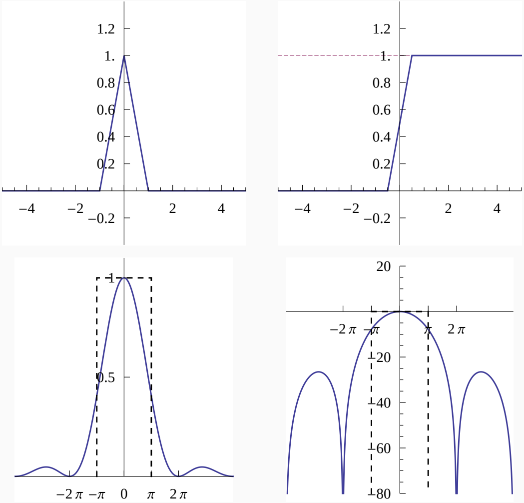
Blue noise



Box

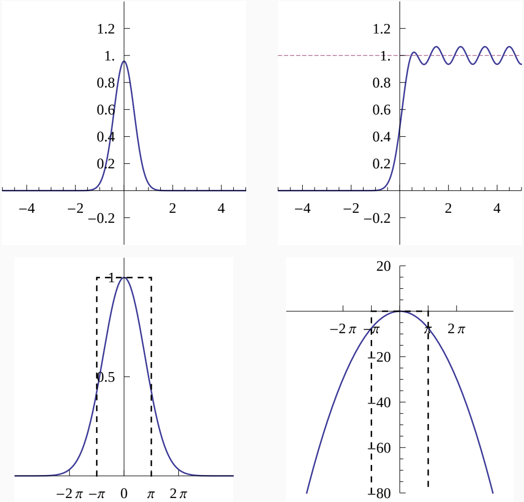


Linear



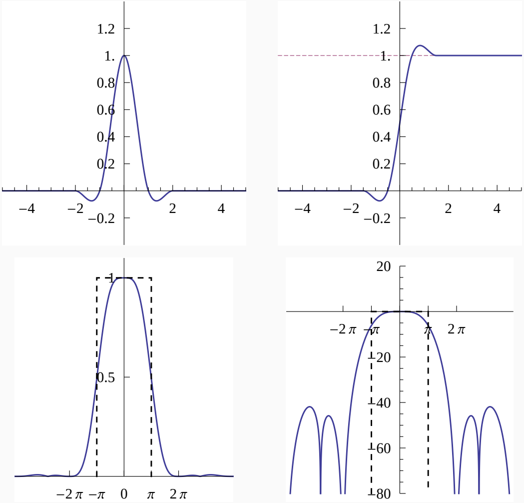
Better anti-aliasing kernels

Gaussian



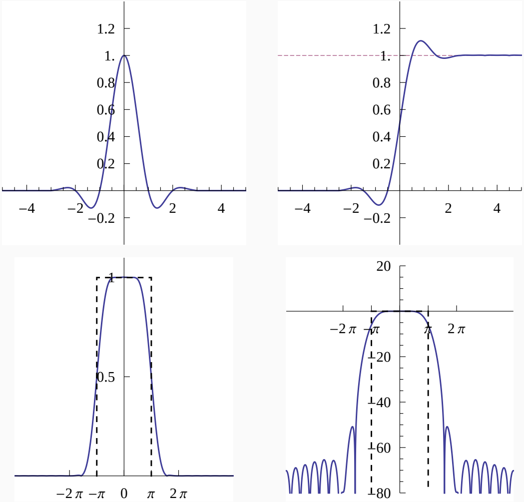
Better anti-aliasing kernels

Keys



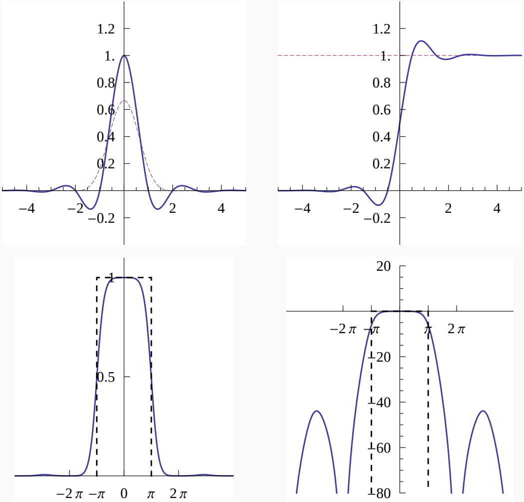
Better anti-aliasing kernels

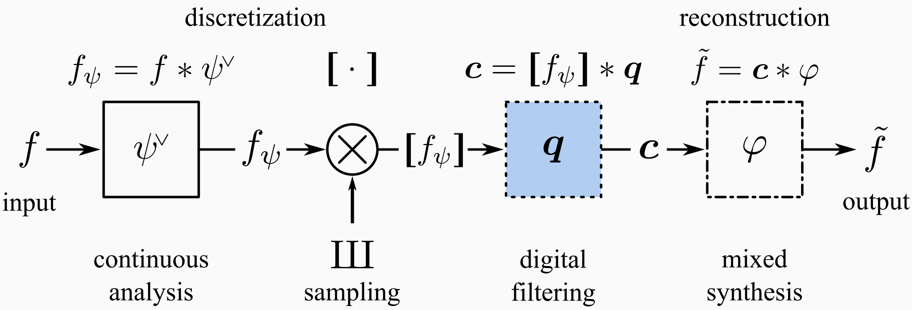
Lanczos

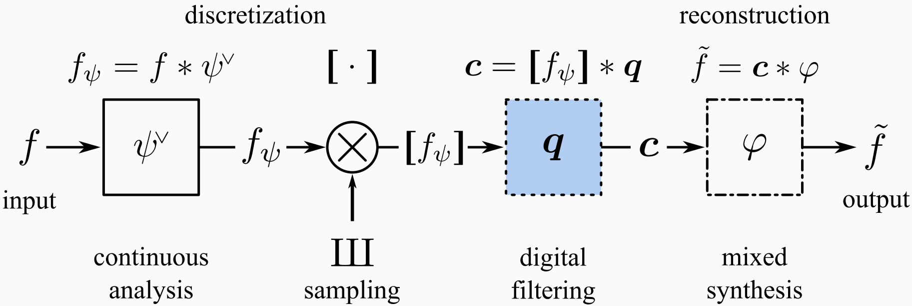


Better anti-aliasing kernels

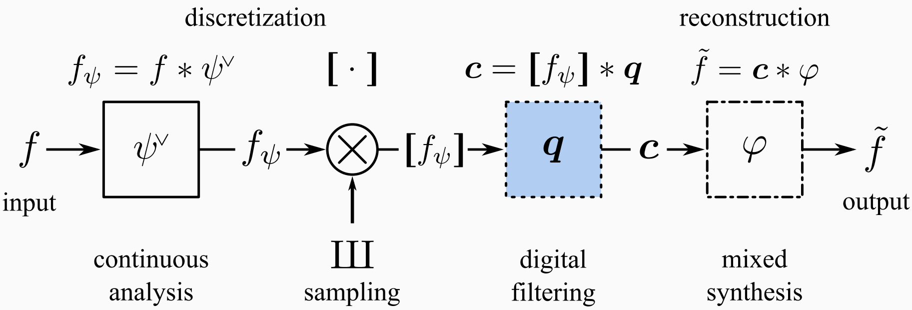
Cardinal B-spline




discretization
reconstruction
$f_\psi = f * \psi^\vee$
$[\,\cdot\,]$
$\boldsymbol{c} = [f_\psi] * \boldsymbol{q}$
$\tilde{f} = \boldsymbol{c} * \varphi$
f
$\psi^\vee$
f_ψ
$[f_\psi]$
$\boldsymbol{q}$
$\boldsymbol{c}$
φ
$\tilde{f}$
input
output
continuous analysis
Ш
sampling
digital filtering
mixed synthesis



Cardinal cubic B-spline



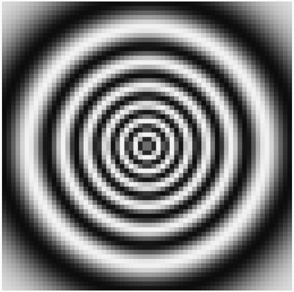
Cardinal cubic B-spline

Needs sample sharing for variance reduction and speed

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Box upsampling





Linear upsampling

Cardinal Cubic B-spline upsampling

Texturing

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Reconstruction is easy, prefiltering is difficult

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Reconstruction is easy, prefiltering is difficult

Non-uniform resampling

- Reconstruct when locally upsampling
- Prefilter when locally downsampling

Texturing

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Reconstruction is easy, prefiltering is difficult

Non-uniform resampling

- Reconstruct when locally upsampling
- Prefilter when locally downsampling
- Jacobian of map from screen to texture coordinates decides

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Reconstruction is easy, prefiltering is difficult

Non-uniform resampling

- Reconstruct when locally upsampling
- Prefilter when locally downsampling
- Jacobian of map from screen to texture coordinates decides

Approximate solution for isotropic downsampling: *Mipmaps*

Texturing

Assuming good reconstruction and prefilter kernels,

- Upsampling needs only reconstruction
- Downsampling needs only prefiltering

Reconstruction is easy, prefiltering is difficult

Non-uniform resampling

- Reconstruct when locally upsampling
- Prefilter when locally downsampling
- Jacobian of map from screen to texture coordinates decides

Approximate solution for isotropic downsampling: *Mipmaps*

Otherwise, use *anisotropic filtering*

References

E. C. Anderson. Monte carlo methods and importance sampling. UC Berkeley, 1999. Lecture notes for Stat 578C.

T. Duff. Polygon scan conversion by exact convolution. In Jacques André and Roger D. Hersch, editors, *Raster Imaging and Digital Typography*, pages 154–168. Cambridge University Press, 1989.

J. Manson and S. Schaefer. Analytic rasterization of curves with polynomial filters. *Computer Graphics Forum (Proceedings of Eurographics)*, 32(2pt4):499–507, 2013.

D. Nehab and H. Hoppe. Random-access rendering of general vector graphics. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(5):135, 2008.

D. Nehab and H. Hoppe. A fresh look at generalized sampling. *Foundations and Trends in Computer Graphics and Vision*, 8(1):1–84, 2014.