2D COMPUTER GRAPHICS

Diego Nehab
Summer 2019
IMPA
ANTI-ALIASING AND TEXTURE MAPPING
Let $f$ be a function and $\psi$ an anti-aliasing filter.
Let $f$ be a function and $\psi$ an anti-aliasing filter.

Value of pixel $p_i$ is given by

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

Let $f$ be a function and $\psi$ an anti-aliasing filter.

Value of pixel $p_i$ is given by

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

How to compute the integral when $f$ is a vector graphics illustration?
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
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 clipping 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 clipping 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 clipping 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?
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 clipping 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 clipping 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?
Assume path $P_i$ with constant color $f_i, \alpha_{f_i}$
Assume path $P_i$ with constant color $f_i, \alpha_{f_i}$
Assume blending over the background $b_i, \alpha_{b_i}$
Assume path $P_i$ with constant color $f_i, \alpha_{f_i}$
Assume blending over the background $b_i, \alpha_{b_i}$
Assume anti-aliasing filter $\psi$ with support $\Omega$
Assume path $P_i$ with constant color $f_i, \alpha_{f_i}$
Assume blending over the background $b_i, \alpha_{b_i}$
Assume anti-aliasing filter $\psi$ with support $\Omega$
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 $f_i, \alpha_f_i$

Assume blending over the background $b_i, \alpha_{b_i}$

Assume anti-aliasing filter $\psi$ with support $\Omega$

Define the coverage $o$ of $P_i$ at pixel $p$

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

The new background $b_{i+1}, \alpha_{i+1}$ 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 \textit{correlated mattes} problem
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

Notice the change in thickness. Notice the change in thickness.
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]

\[
b_{i+1}, \alpha_{i+1} = \gamma(\gamma^{-1}(f_i, \alpha_i \oplus b_i, \alpha_i) \cdot (1 - o) + \gamma^{-1}(b_i, \alpha_i) \cdot o)
\]
A random variable $X$ is a function that maps outcomes to numbers.

The associated cumulative distribution function $F_X(a) = P[X \leq a]$ i.e., it measures the probability that the numerical value is at most $a$.

The associated probability density function $f_X(t)$ 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 $\mu_X$) is $E[X] = \int_{-\infty}^{\infty} t \, f_X(t) \, dt = \mu_X$. (1) i.e., the mean value weighted by the probability density function.
A random variable $X$ is a function that maps outcomes to numbers. The associated cumulative distribution function $F_X$ is such that

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

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

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

i.e., the mean value weighted by the probability density function.
The associated variance $\text{var}(X) = \sigma_X^2$ is

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

and the standard deviation is $\sigma_X$. 
The associated variance $\text{var}(X) = \sigma_X^2$ is

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

and the standard deviation is $\sigma_X$.

Measure how much the random variable deviates from the mean.
The associated variance \( \text{var}(X) = \sigma_X^2 \) is
\[
\text{var}(X) = E[(X - \mu_X)^2] = E[X^2] - E^2[X] = \sigma_X^2
\]
and the standard deviation is \( \sigma_X \).

Measure how much the random variable deviates from the mean.

The sample average is
\[
\bar{X}_n = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)
\]
The associated variance \( \text{var}(X) = \sigma_X^2 \) is

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

and the standard deviation is \( \sigma_X \).

Measure how much the random variable deviates from the mean.

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

Law of large numbers

\( \bar{X}_n \rightarrow \mu_X \) for \( n \rightarrow \infty \)
The associated variance $\text{var}(X) = \sigma_X^2$ is

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

and the standard deviation is $\sigma_X$.

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

$$\text{var}(\overline{X}_n) = \text{var} \left( \frac{1}{n} \sum X_i \right) = \frac{1}{n^2} \sum \text{var}(X_i) = \frac{\sigma_X^2}{n}$$
Monte Carlo integration

Start by expressing an integral as the expectation of a random variable
Estimate expectation by sample mean
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 $\Omega$

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

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

This is the basis of supersampling

The solution to our anti-aliasing problems
Let $g : \mathbb{R}^2 \rightarrow RGB$ map positions to linear color

Consider an \textit{anti-aliasing kernel} $\psi$
Let $g : \mathbb{R}^2 \rightarrow \text{RGB}$ map positions to linear color.

Consider an anti-aliasing kernel $\psi$.

The linear color at pixel $p$ is

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

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

Consider an anti-aliasing kernel $\psi$

The linear color at pixel $p$ is

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

$$= E \left[ \frac{g(p - X) \psi(X)}{f_X(X)} \right]$$
Let $g : \mathbb{R}^2 \to RGB$ map positions to linear color

Consider an anti-aliasing kernel $\psi$

The linear color at pixel $p$ is

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

$$= E \left[ \frac{g(p - X) \psi(X)}{f_X(X)} \right]$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \frac{g(p - X_i) \psi(X_i)}{f_X(X_i)}$$
Let $g : \mathbb{R}^2 \to \text{RGB}$ map positions to linear color

Consider an anti-aliasing kernel $\psi$

The linear color at pixel $p$ is

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

$$= E \left[ \frac{g(p - X) \psi(X)}{f_X(X)} \right]$$

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

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

\[
\hat{c}(p) \approx \sum_{i=1}^{n} g(p - X_i) \psi(X_i) f_{X}(X_i)
\]
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*

\[
c(p) \approx \frac{\sum_{i=1}^{n} \frac{g(p - X_i) \psi(X_i)}{f_X(X_i)}}{\sum_{i=1}^{n} \frac{\psi(X_i)}{f_X(X_i)}}
\]
What happens if we choose \( f_X(t) \propto g(t) \)?

\[
\int \Omega g(t) \, dt = E\left[ g(X) f_X(X) \right] = E\left[ \alpha \right] = g(X) f(X)
\]

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

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

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.
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!

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

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

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

Uniform, stratified, low-discrepancy (e.g. Poisson disk, Lloyd relaxation)
Many different point distributions have $f_X = 1/A_\Omega$ in $\Omega$
Uniform, stratified, low-discrepancy (e.g. Poisson disk, Lloyd relaxation)
Variance of $\bar{X}_n$ is not the same for all of them!
16 SAMPLES

Regular
16 SAMPLES

Uniform
16 SAMPLES

Stratified
Blue noise
64 SAMPLES

Regular
64 SAMPLES

Uniform
64 SAMPLES

Stratified
Blue noise
256 SAMPLES

Regular
256 SAMPLES

Uniform
Stratified
256 SAMPLES

Blue noise
1024 SAMPLES

Regular
1024 SAMPLES

Uniform
Stratified
Blue noise
Better anti-aliasing kernels

Box
**Better anti-aliasing kernels**

Linear
Better anti-aliasing kernels

Gaussian
Better anti-aliasing kernels

Keys
Better anti-aliasing kernels

Lanczos
Better anti-aliasing kernels

Cardinal B-spline
Generalized sampling

\[ f_\psi = f \ast \psi^\vee \]

\[ c = [f_\psi] \ast q \]

\[ \tilde{f} = c \ast \varphi \]
Generalized sampling

\[ f_\psi = f \ast \psi^\gamma \]

\[ c = [f_\psi] \ast q \]

\[ \tilde{f} = c \ast \varphi \]

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
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
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
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