CS380: Introduction to Computer Graphics
Sampling
Chapter 16

Min H. Kim
KAIST School of Computing

Texture mapping

SUMMARY

Normal mapping

- The data from a texture can also be interpreted in more interesting ways.
- In normal mapping, the r,g,b values from a texture are interpreted as the three coordinates of the normal at the point.
- This normal data can then be used as part of some material simulation.

Environment cube maps

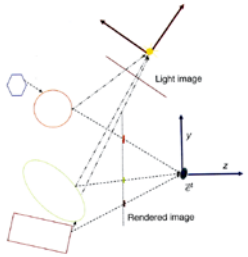
- Textures can also be used to model the environment in the distance around the object being rendered.
- In this case, we typically use 6 square textures representing the faces of a large cube surrounding the scene.

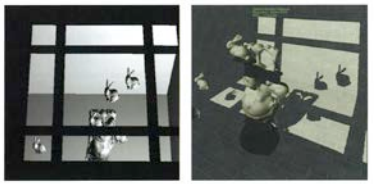
Projector texture mapping

- There are times when we wish to glue our texture onto our triangles using a *projector* model, instead of the affine gluing model.
- For example, we may wish to simulate a slide projector illuminating some triangles in space.

Shadow mapping

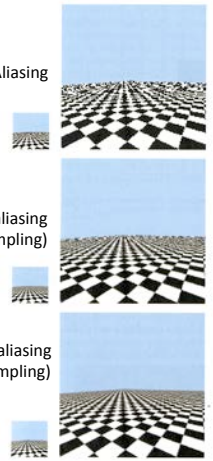
- The idea is to first create and store a z-buffered image from the point of view of the light, and then compare what we see in our view to what the light saw in its view.





Chapter 16

SAMPLING

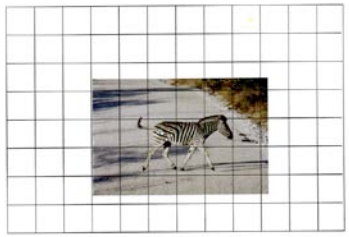


Sampling

- Point sampling: continuous vector → discrete pixel
- Our scenes are described with triangles giving a continuous 2D color field.
- Our images are digital/discrete made up of a grid of dots.
- Need to make a bridge between these two worlds (continuous vs. discrete).
- Else we will get some unnecessary artifacts called “aliasing” artifacts.
  - Jaggies, moire patterns, flickering

Sampling

- These occur when there is too much detail to fit in one pixel.
- We can mitigate these artifacts by averaging up the colors within a pixel's square.
- This is called *anti-aliasing.*



Two models

- A *continuous image,* $I(x_w, y_w)$, is a bivariate function.
  - range is a linear color space.
- A *discrete image* I[i][j] is a two dimensional array of color values.
- We associate each pair of integers i, j, with the continuous image coordinates $x_w = i$ and $y_w = j$

Aliasing

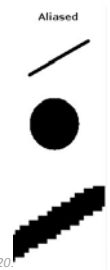
- The simplest and most obvious method to go from a continuous to a discrete image is by *point sampling.*
- To obtain the value of a pixel i, j, we sample the continuous image function at a single integer valued domain location:

$$\mathrm{I[i][j]} \leftarrow I(i, j)$$

- This can results in unwanted artifacts.

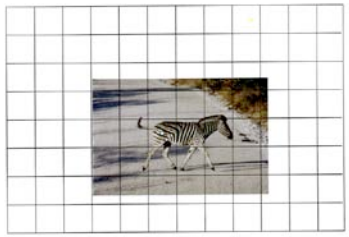
Aliasing

- Scene made up of black and white triangles: jaggies at boundaries
  - Jaggies will crawl during motion
- If triangles are small enough then we get random values or weird patterns.
  - Jaggies will crawl during motion



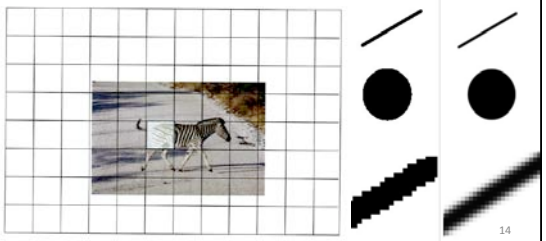
Aliasing

- If triangles are small enough then we get random values or weird patterns
  - Will flicker during motion
- The heart of the problem: too much information in one pixel



Anti-aliasing

- Intuitively: the single sample is a bad value, we would be better off setting the pixel value using some kind of average value over some appropriate region.
- In the above examples, perhaps some gray value.



Anti-aliasing

- Mathematically this can be modeled using *Fourier analysis*.
  - Breaks up the data by "frequencies" and figures out what to do with the un-representable high frequencies.

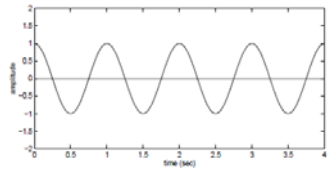
Nyquist Sampling Theorem

- It states the following:
  - "The sampling frequency should be at least twice the highest frequency contained in the signal."

$$f_s \geq 2 f_c$$

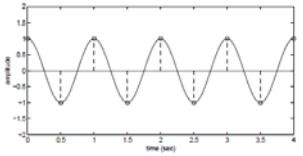
Example

- An input signal with f = 1 Hz

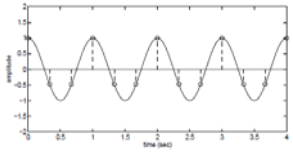


Example

- If we sample by more than 2 Hz, we can reconstruct the shape correctly



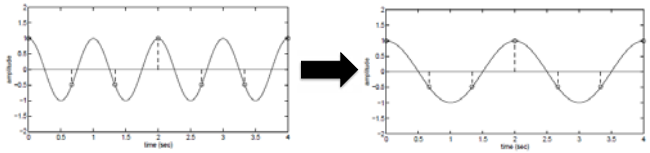
2 Hz



3 Hz

Example

- If we sample by 1.5 Hz (<= 2 Hz), there might be an ambiguity about the signal shape.



1.5 Hz

Anti-aliasing

- We can also model this as an optimization problem.
- These approaches lead to:

$$\mathrm{I[i][j]} \leftarrow \iint_{\Omega} I(x,y)F_{i,j}(x,y)dxdy$$

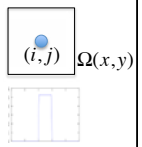
- where $F_{i,j}(x,y)$ is some function that tells us how strongly the continuous image value at $[x,y]^t$ should influence the pixel value $\mathrm{i,j}$

Anti-aliasing

- In this setting, the function $F_{i,j}(x,y)$ is called *a filter*.
  - In other words, the best pixel value is determined by performing some continuous weighted averaging near the pixel's location.
  - Effectively, this is like blurring the continuous image before point sampling it.

Box filter

- We often choose the filters $F_{i,j}(x,y)$ to be something non-optimal, but that can more easily computed with.
- The simplest such choice is a *box filter*, where $F_{i,j}(x,y)$ is zero everywhere except over the 1-by-1 square center at $x = i, y = j$.

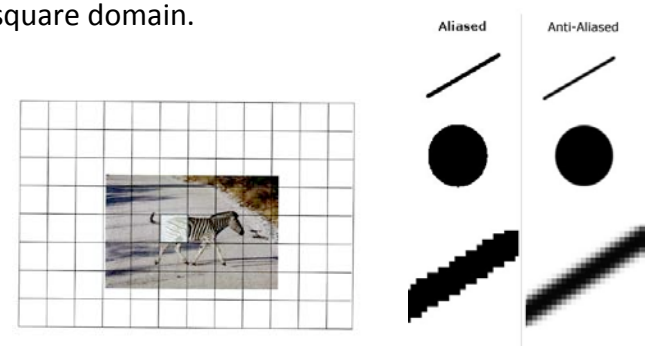


- Calling this square $\Omega_{i,j}$, we arrive at

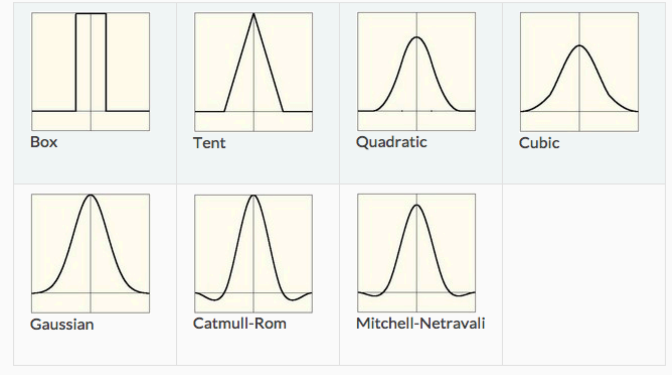
$$\mathrm{I[i][j]} \leftarrow \iint_{\Omega_{i,j}} I(x,y)dxdy$$

Box filter

- In this case, the desired pixel value is simply the average of the continuous image over the pixel's square domain.



Filters



Over-sampling

- Even that integral is not really reasonable to compute
- Instead, it is approximated by some sum of the form:

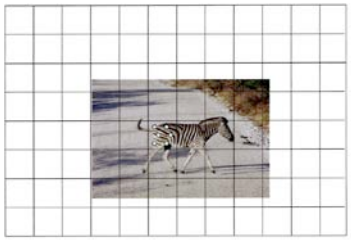
$$\mathrm{I[i][j]} \leftarrow \frac{1}{n}\sum_{k=1}^{n} I(x_k, y_k)$$

where *k* indexes some set of locations (x_k, y_k) called the sample locations.

- The renderer first produces a "high resolution" color and z-buffer "image",
  - where we will use the term *sample* to refer to each of these high resolution pixels.

Over-sampling

- Then, once rasterization is complete, groups of these samples are averaged together, to create the final lower resolution image.



Super-sampling

- If the sample locations for the high resolution image form a regular, high resolution grid, then this is called *super sampling.*
- We can also choose other sampling patterns for the high resolution "image",
  - Such less regular patterns can help us avoid systematic errors that can arise when using the sum to replace the integral.

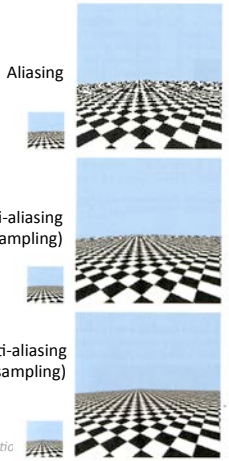
Multi-sampling

- Render to a "high resolution" color and z-buffer
- *During the rasterization* of each triangle, "coverage" and z-values are computed at this sample level.
- But for efficiency, the fragment shader is only called **only once per final resolution pixel.**
  - This color data is shared between all of the samples hit by the triangle in a single (final resolution) pixel.
- Once rasterization is complete, groups of these high resolution samples are averaged together.

Multi-sampling

- Multisampling can be an effective anti-aliasing method since, without texture mapping, colors tend to vary quite slowly over each triangle, and thus they do not need to be computed at high spatial resolution.
- To deal with aliasing that occurs during texture mapping, we have the advantage of possessing the texture image in hand at the outset of the rendering process.
- This leads to specialized techniques such as *mip mapping*.

Aliasing vs. anti-aliasing

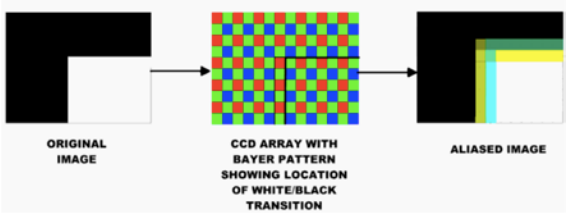


Camera

- In digital cameras, anti-aliasing is accomplished by a combination of the spatial integration that happens over the extent of each pixel sensor, as well as by the optical blurring that happens at due to the lens. Some cameras also include additional optical elements specifically to blur the continuous image data before it is sampled at the sensors.

Camera: Demosaicing problem

- Imagine a black-on-white corner
- Color aliasing happens



Camera: Foveon Sensor

- Transparent layers of RGB
- No demosaicing → no moiré artifact

Human vision

- In human vision, aliasing artifacts are not typically encountered.
  - Most of the anti-aliasing, at least in the foveal (central) region of vision, is due to the optical blurring of light (Airy disk), which happens well before it hits the receptor cells.
  - The irregular spatial layout of the sensor cells in the retina also helps by effectively providing spatial jitter (randomness) which turns noticeable aliasing into less conspicuous noise.

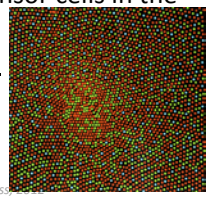
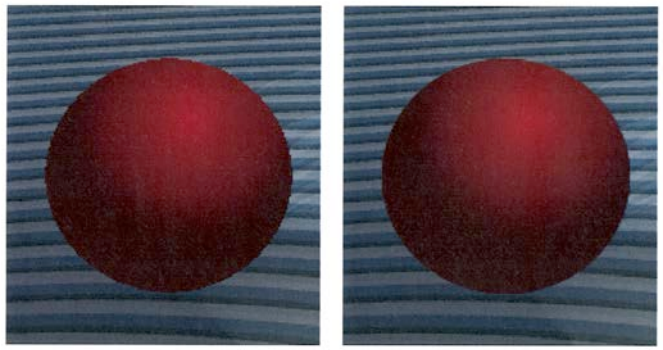


Image compositing

- Given two discrete images, a foreground, $\mathbf{I}^f$, and background, $\mathbf{I}^b$, that we want to combine into one image $\mathbf{I}^c$.
- Simple: in composite, use foreground pixels where they are defined. Else use background pixels.
- This will give us a jagged boundary.
- Real image would have "boundary" pixels with blended colors.
- But this requires using "sub-pixel" information.

Image compositing



Alpha blending

- Associate with each pixel in each image layer, a value, $\alpha[i][j]$, that describes the overall *opacity* or *coverage* of the image layer at that pixel.
  - An alpha value of 1 represents a fully opaque/ occupied pixel, while a value of 0 represents a fully transparent/empty one.
  - A fractional value represents a partially transparent (partially occupied) pixel.
- Alpha will be used during compositing.

Alpha definition

- More specifically, let $I(x,y)$ be a continuous image, and let $C(x,y)$ be a binary valued *coverage function* over the continuous (x,y) domain, with a value of 1 at any point where the image is "occupied" and 0 where it is not.
- Let us store in our discrete image the values:

$$\mathrm{I}[i][j] \leftarrow \iint_{\Omega_{i,j}} I(x,y)C(x,y)dxdy$$

$$\alpha[i][j] \leftarrow \iint_{\Omega_{i,j}} C(x,y)\,dx\,dy$$

Over operation

- To compose $\mathrm{I}^{f}[i][j]$ over $\mathrm{I}^{b}[i][j]$, we compute the composite image colors, $\mathrm{I}^{c}[i][j]$, using

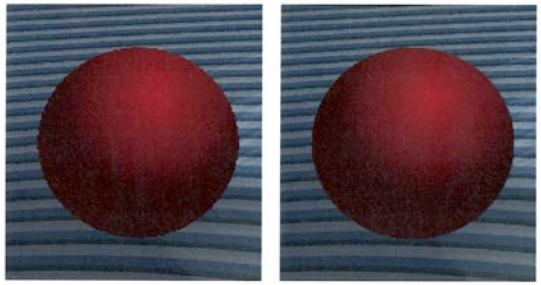
  $\mathrm{I}^{c}[i][j] \leftarrow \mathrm{I}^{f}[i][j] + \mathrm{I}^{b}[i][j]\,(1 - \alpha^{f}[i][j])$

  That is, the amount of observed background color at a pixel is proportional to the transparency of the foreground layer at that pixel.
- Likewise, alpha for the composite image can be computed as:

$$\alpha^{c}[i][j] \leftarrow \alpha^{f}[i][j] + \alpha^{b}[i][j]\,(1 - \alpha^{f}[i][j])$$

Over operation

- If background is opaque, so the composite pixel is opaque.
- But we can model more general case as part of blending multiple layers.



Over properties

- This provides a reasonable approximation to the correctly rendered image.
- One can easily verify that the over operation is associative but not commutative. That is,

$$I^a \text{ over } (I^b \text{ over } I^c) = (I^a \text{ over } I^b) \text{ over } I^c$$

but, $$I^a \text{ over } I^b \neq I^b \text{ over } I^a$$

- NB pulling mattes from real images against controlled or esp. uncontrolled backgrounds is an area of research.

Alpha blending error

- Continuous composition

$$I^c[i][j] \leftarrow \iint_{\Omega_{i,j}} I^f(x,y)C^f(x,y) + I^b(x,y)C^b(x,y) - I^b(x,y)C^b(x,y)C^f(x,y)\,dx\,dy$$

$$= \iint_{\Omega_{i,j}} I^f(x,y)C^f(x,y)\,dx\,dy + \iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)\,dx\,dy - \iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)C^f(x,y)\,dx\,dy$$

Alpha blending error

- Alpha blending

$$I^c[i][j] \leftarrow I^f[i][j] + I^b[i][j]\,(1 - \alpha^f[i][j])$$

$$= \iint_{\Omega_{i,j}} I^f(x,y)C^f(x,y)\,dx\,dy + \left(\iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)\,dx\,dy\right)\left(1 - \iint_{\Omega_{i,j}} C^f(x,y)\,dx\,dy\right)$$

$$= \iint_{\Omega_{i,j}} I^f(x,y)C^f(x,y)\,dx\,dy + \iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)\,dx\,dy - \iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)\,dx\,dy \cdot \iint_{\Omega_{i,j}} C^f(x,y)\,dx\,dy$$

Alpha blending error

- The difference between two

$$\iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)C^f(x,y)\,dx\,dy - \iint_{\Omega_{i,j}} I^b(x,y)C^b(x,y)\,dx\,dy \times \iint_{\Omega_{i,j}} C^f(x,y)\,dx\,dy$$

- Therefore the error is the difference between the integral of a product and a product of integrals.
- The amount of "correlation" between the distribution of foreground coverage in some pixel and the distribution of the background data within that pixel.