PBRT: Texture

Tuesday, January 5, 2021 9:50 PM

10 Texture

The spatial variation of material properties is described by textures.

A ***texture*** is a ***function*** $T(p)$ that maps points in some domain to values in some other domain:

- Example domains: a surface's (u, v) ***parametric space*** or (x, y, z) ***object space***.
- Example codomains: spectra (like Kd in MatteMaterial and PlasticMaterial) or real numbers (like roughness in MatteMaterial and PlasticMaterial).

Textures introduce high-frequency content into the image function. Aliasing. Increasing the sampling rate and nonuniform sampling are 2 alternative antialiasing techniques for textures, but PBRT ***does something else***: texture functions adjust their frequency content (remove high-frequency detail) based on the rate at which they are being sampled (the image sampling rate); this is a form of ***prefiltering***.

10.1 Sampling and antialiasing

When sampling the ***scene***: The infinite frequency content of geometric edges and hard shadows guarantees aliasing, no matter how high the sampling rate. We lessen their effect with noise caused by nonuniform sampling.

When sampling a ***texture function***: If the texture function has an analytic form, we may design it or change it so that it doesn't have high-frequency content.

10.1.1 Finding the texture sampling rate

The texture function $T(p)$ is defined on a surface in the scene. But when rendering the surface, sampling (ray shooting) takes place through points with coordinates in the image plane (pixel coordinates). So we need to map points on the image plane to points on the texture that correspond to points on the surface in the scene: $T(f(x, y))$, where f maps image points (x, y) to points (s, t) on the texture (are s and t the tangent vectors as in shading space? *"In pbrt, we will use the convention that 2D texture coordinates are denoted by (s, t); this helps make clear the distinction between the intrinsic (u, v) parameterization of the underlying surface and the (possibly different) coordinate values used for texturing."*).

(x_r, y_r) denotes the ***resolution*** of the image. $(\frac{1}{x_r}, \frac{1}{x_y})$ is a sample spacing of 1 pixel, or 1 sample per pixel.

To remove high-frequency content from $T(p)$, first we want to establish a relationship between pixel coordinates (x, y) and texture coordinates (s, t), as well as the relationship between their sampling rates: we don't want the texture sampling rate to be higher than the image sampling rate.

Computing $f(x, y)$ is hard: it depends on geometry, camera projection, and mappings to texture coordinates. But we don't need it! We just need its differential changes in the x and y ***pixel directions***:

$\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$

where the partial derivatives express the change in (s, t) texture coordinate caused by a change in (x, y) ***discrete pixel*** coordinate.

With $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$, we can approximate f:

$$f(x', y') \approx f(x, y) + (x' - x)\frac{\partial f}{\partial x} + (y' - y)\frac{\partial f}{\partial y}$$

which is a reasonable approximation when the partial derivatives change slowly relative to $(x' - x)$ and $(y' - y)$. But note that it depends on $f(x, y)$ and we don't know it. But it doesn't matter, because we are interested in $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ and the conditions under which they make for a good approximation (that they change slowly with respect to $(x' - x)$ and $(y' - y)$).

Because f outputs a (s, t) texture coordinate, $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ directly yield the ***texture sampling rate that corresponds to a given image sampling rate***: if the (x, y) image coordinate changes by $(\partial x, \partial y)$, then the texture coordinate changes by $\partial f = (\partial s, \partial t)$.

Other partial derivatives:

- $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$, partial derivatives of P: $(x, y) \rightarrow (wp_x, wp_y, wp_z)$, the mapping from image space to world space.
- $\frac{\partial u}{dx}$ and $\frac{\partial v}{\partial y}$, partial derivatives of U: (x, y) → u and V: $(x, y) \rightarrow v$, the mapping from image space to ***UV parametric space***.

For $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$ to make sense, we need to assume that the surface is ***locally flat*** (a concept analog to local linearity?). This assumption is reasonable for small sampling differentials.

These partial derivatives are computed using RayDifferentials. RayDifferentials have a main ray r that intersects the surface at point p. Assuming that the surface is locally flat, we can represent the local surface geometry as a plane that is tangent to the surface at p:

$$ax + by + cz + d = 0$$

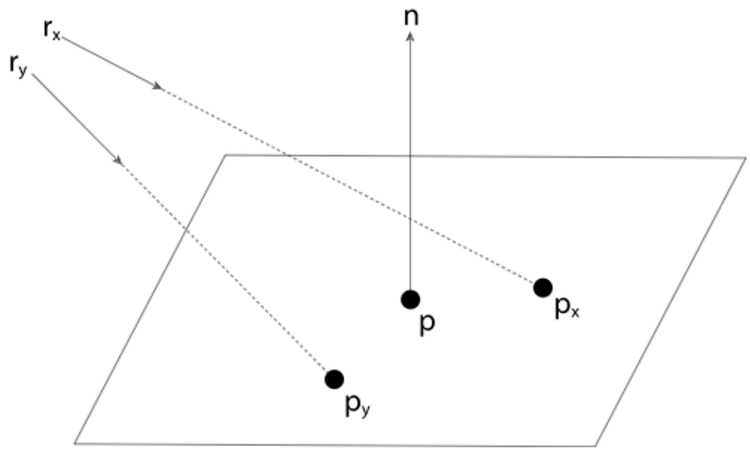
where $a = n_x$, $b = n_y$, $c = n_z$, and $d = -(n \cdot p)$.

The ray differentials r_x and r_y intersect this local surface geometry at p_x and p_y, which are used to approximate the partial derivatives:

$$\frac{\partial p}{\partial x} \approx p_x - p$$

$$\frac{\partial p}{\partial y} \approx p_y - p$$

∂x and ∂y are 1-pixel changes. So a change of 1 pixel in the image coordinate in the x direction causes a change of $p_x - p$, etc.



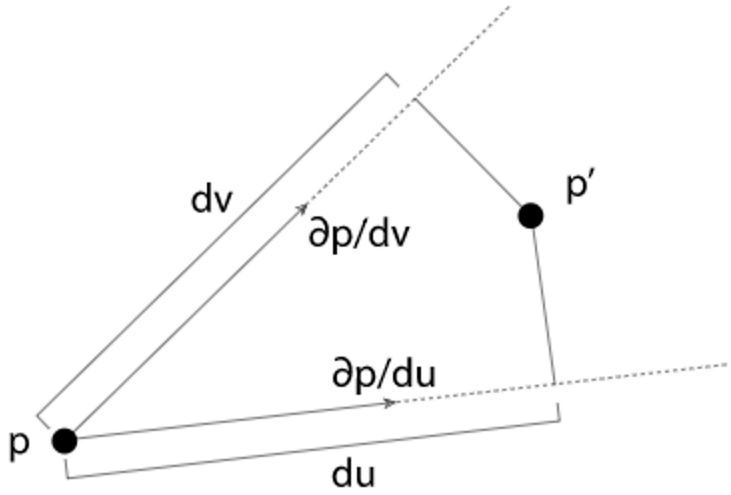
To find the RayDifferential-plane intersection, we find the t parameter ray differential that yields that poine. We use this equation:

$$t = \frac{-((a,b,c)\cdot \boldsymbol{o}) - d}{(a,b,c)\cdot \boldsymbol{d}}$$

where bold $\boldsymbol{d}$ is the ray differential's direction and plain d is from the plane's equation.

Now that p_x and p_y are known, we can find their respective coordinates in UV parametric space.

We start by setting up a 2D coordinate system on the local surface approximation plane, with p as its origin and $\{\frac{\partial p}{\partial u}, \frac{\partial p}{\partial v}\}$ as its (possibly nonorthogonal) basis.



Now we want to find the ***coordinate vectors*** of p_x and p_y relative to this basis, that is, as linear combinations of the $\{\frac{\partial p}{\partial u}, \frac{\partial p}{\partial v}\}$ basis vectors per the unique representation theorem:

$[p']_{uv} = \begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix}$ in UV space, such that $p' = p + \Delta_u \frac{\partial p}{\partial u} + \Delta_v \frac{\partial p}{\partial v}$ in camera space, where p' is p_x or p_y or any other point on the local surface approximation plane.

The ***matrix equation form*** $Ax = b$ of the ***vector equation*** $p' - p = \Delta_u \frac{\partial p}{\partial u} + \Delta_v \frac{\partial p}{\partial v}$ is:

$$\begin{bmatrix} \frac{\partial p_x}{\partial_u} & \frac{\partial p_x}{\partial_v} \\ \frac{\partial p_y}{\partial_u} & \frac{\partial p_y}{\partial_v} \\ \frac{\partial p_z}{\partial_u} & \frac{\partial p_z}{\partial_v} \end{bmatrix} \begin{bmatrix} \Delta_u \\ \Delta_V \end{bmatrix} = \begin{bmatrix} p'_x - p_x \\ p'_y - p_y \\ p'_z - p_z \end{bmatrix}$$

The SurfaceInteraction object is given $\frac{\partial \mathrm{p}}{\partial \mathrm{u}}$ and $\frac{\partial p}{\partial_v}$ during construction: SurfaceInteraction::dpdu and SurfaceInteraction dpdv.

- [ ] TODO: who computes dpdu and dpdv and passes them to SurfaceInteraction?

Since $\frac{\partial \mathrm{p}}{\partial \mathrm{u}}$ and $\frac{\partial p}{\partial_v}$ and p and p_x and p_y are known, we can find $\begin{bmatrix} \Delta_u \\ \Delta_V \end{bmatrix}$ by solving the matrix equation.

$\Delta_u = (\frac{du}{dx}, \frac{du}{dy})$ and $\Delta_V = (\frac{dv}{dx}, \frac{dv}{dy})$ in code.

- [ ] Image space is screen space or raster space?

10.1.2 Filtering texture functions

3 ways of filtering texture functions, with the objective of ***band-limiting*** the function, that is, removing frequencies that are higher than the Nyquist limit:

- Box filtering yields good results for many texture functions. The box filter has the effect of averaging the samples taken for a given region.
- ***Clamping*** replaces high-frequency terms with their averages.
- ***Supersampling*** samples and filters the function at multiple locations near the main sample. If the chosen filter is the box filter, this has the effect of averaging the function near the main sample point.

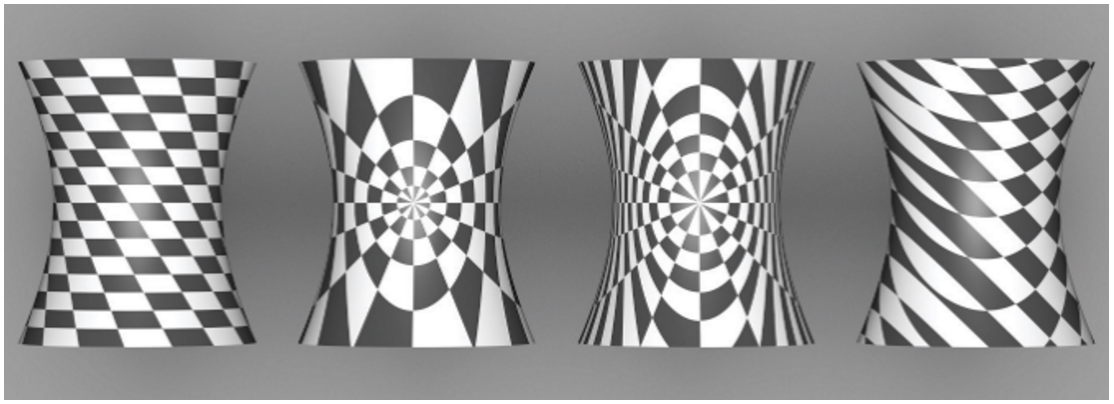
10.2 Texture coordinate generation

Parameterizations of a surface.

One goal is to avoid distortion.

Texture mappings compute texture coordinates, which may be 2D or 3D.

(u, v) mapping, spherical mapping, cylindrical mapping, and planar mapping



(u, v) mapping is just one type of surface parameterization for mapping points on the surface to points on the texture. Other types are ***spherical mapping***, ***cylindrical mapping***, and ***planar mapping***.

10.2.1 2D (u, v) mapping

The (u, v) coordinate of an surface intersection point p is ***known somehow***.

The (u, v) coordinate is mapped to an (s, t) coordinate, where the (s, t) coordinate exists in shading space, on the plane that is tangent to the intersection point.

The (u, v) coordinate is scaled and shifted by (c_u, c_v) and (d_u, d_v), which are scalars associated to the Mapping object. That's the mapping: scale and shift.

$T: (u, v) \rightarrow (s, t) = (c_u u + d_u, c_v v + d_v)$

So the ***derivative of this vector-valued function*** is the vector:

$$\left[\left(\frac{\partial s}{\partial u}, \frac{\partial s}{\partial v}\right) \quad \left(\frac{\partial t}{\partial u}, \frac{\partial t}{\partial v}\right)\right]$$

where $s = c_u u + d_u$ and $t = c_v v + d_v$ according to the T mapping. And:

$$\frac{\partial s}{\partial u} = c_u$$

$\frac{\partial s}{\partial v} = 0$, because s does not depend on v

$\frac{\partial t}{\partial u} = 0$, because t does not depend on u

$$\frac{\partial t}{\partial v} = c_v$$

As before, we want to tie the texture function sampling rate to the image function sampling rate. So u and v are functions of $p = (x, y)$:

$u(p) = u(x, y)$ and $v(p) = v(u, x)$

And T is a ***composition***:

$T: (u(x, y), v(x, y)) \rightarrow (s, t) = (c_u u(x, y) + d_u, c_v v(x, y) + d_v)$

And the partial derivatives according to the ***chain rule*** are:

$\frac{\partial s}{dx} = \frac{\partial u}{\partial x}\frac{\partial s}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial s}{\partial v}$ $\frac{\partial t}{dx} = \frac{\partial u}{\partial x}\frac{\partial t}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial t}{\partial v}$	*a change in pixel coordinate x that causes a change in UV coordinate* (u, v) *that causes a change in shading coordinate* (s, t)
$\frac{\partial s}{dy} = \frac{\partial u}{\partial y}\frac{\partial s}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial s}{\partial v}$ $\frac{\partial t}{dy} = \frac{\partial u}{\partial y}\frac{\partial t}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial t}{\partial v}$	*a change in pixel coordinate y that causes a change in UV coordinate* (u, v) *that causes a change in shading coordinate* (s, t)

Making substitutions:

$$\frac{\partial s}{\partial x} = \frac{\partial u}{\partial x}c_u + \frac{\partial v}{\partial x}0 = \frac{\partial u}{\partial x}c_u$$

$$\frac{\partial t}{\partial x} = \frac{\partial u}{\partial x}0 + \frac{\partial v}{\partial x}c_v = \frac{\partial v}{\partial x}c_v$$

$$\frac{\partial s}{\partial y} = \frac{\partial u}{\partial y}c_u + \frac{\partial v}{\partial y}0 = \frac{\partial u}{\partial y}c_u$$

$$\frac{\partial t}{\partial y} = \frac{\partial u}{\partial y}0 + \frac{\partial v}{\partial y}c_v = \frac{\partial v}{\partial y}c_v$$

10.2.2 Spherical mapping

- [ ] TODO

10.2.3 Cylindrical mapping

- [ ] TODO

10.2.4 Planar mapping

- [ ] TODO

10.2.5 3D mapping

- [ ] TODO

10.3 Texture interface and basic textures

The Texture template class can be parameterized with Float (e.g. roughness values) and Spectrum (e.g. reflectances) types.

10.3.1 Constant texture

Useful for diffuse materials that are not spatially-varying.

10.3.2 Scale texture

- [ ] TODO

10.3.3 Mix texture

- [ ] TODO

10.3.4 Bilinear interpolation

- [ ] TODO

10.4 Image texture

- [ ] TODO

10.5 Solid and procedural texturing

- [ ] TODO

10.6 Noise

- [ ] TODO