Sampling (continued) & Textures

UCSD CSE 168
Rendering
Tzu-Mao Li
Last time: sampling theorem

spatial domain $\star$ Fourier transform $\rightarrow$ frequency domain

$\mathcal{F}$
What if we have randomly placed samples?
What if we have randomly placed samples?

aliasing becomes noise!
Fourier transform of random sampling on expectation a uniform distribution "white noise"
Quiz: which sampling distribution is better?
Sampling distribution of human eyes

quiz 1: how is it different from our displays?
quiz 2: how is it different from independent random sampling?
Human eyes’ sampling follow a Poisson-disc/blue-noise distribution for each disk of distance $r$ at each point, there is no other points in the disk.
Fourier transform of Poisson-disk/blue-noise sampling

5.3.3 Tiling-based methods

There are some tile-based approaches that can be used to generate blue noise samples. Tile-based methods overcome the computational complexity of dart-throwing and/or relaxation based approaches in generating blue noise sampling patterns. In computer graphics community, two tile-based approaches are well known: First approach uses a set of precomputed tiles, with each tile composed of multiple samples, and later use these tiles, in a sophisticated way, to pave the sampling domain. Second approach employed tiles with one sample per tile and uses some relaxation-based schemes, with look-up tables, to improve the overall quality of samples.

Although many blue noise sample generation algorithms exist, none of them are easily extendable to higher dimensions (\( \geq 3 \)).

5.4 Interpreting and exploiting knowledge of the sampling spectra

Recently, it has been shown that the low frequency region of the radial power spectrum (of a given sampling pattern) plays a crucial role in deciding the overall variance convergence rates of sampling patterns used for Monte Carlo integration. Since blue noise sampling patterns contains almost no radial energy in the low frequency region, they are of great interest for future research to obtain fast results in rendering problems. Surprisingly, Poisson Disk samples have shown the convergence rate of \( O(N^{1/2}) \) which is the same as given by purely random samples. This can be explained by looking at the low frequency region in the radial power spectrum of Poisson Disk samples (Fig. 5.9) which is not zero. The importance of the shape of the radial mean power spectrum in the low frequency region demands methods and algorithms that could eventually allow sample generation directly from a target Fourier spectrum.

5.4.1 Radially-averaged periodograms

Figures 5.6, 5.8 and 5.9 depict radially averaged periodograms of the various sampling strategies described in this chapter. These spectra reveal two important characteristics of estimators built using the corresponding sampling strategies.
5.4 Interpreting and exploiting knowledge of the sampling spectra

Recently [39], it has been shown that the low frequency region of the radial power spectrum (of a given sampling pattern) plays a crucial role in deciding the overall variance convergence rates of sampling patterns used for Monte Carlo integration. Since blue noise sampling patterns contain almost no radial energy in the low frequency region, they are of great interest for future research to obtain fast results in rendering problems. Surprisingly, Poisson Disk samples have shown the convergence rate of $O(N^{-1})$, which is the same as given by purely random samples. This can be explained by looking at the low frequency region in the radial power spectrum of Poisson Disk samples (Fig. 5.9) which is not zero. The importance of the shape of the radial mean power spectrum in the low frequency region demands methods and algorithms that could eventually allow sample generation directly from a target Fourier spectrum.
Which sampling distribution is better?

- **Human-eyes-like**
- **Independent random sampling**
- **Regular sampling**
Dart throwing for Poisson disc sampling

[Cook 1986]

(fancy animation from Wojciech Jarosz)

Lloyd relaxation for Poisson disc sampling

developed at ~1957, published at 1982

Least Squares Quantization in PCM

video from
Stratified/jittered sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }
Stratified/jittered sampling

Samples (expected) Fourier transform radial mean

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectrum of various sampling patterns. He advocated three important features for an ideal radial power spectrum: First, its peak should be at the samples (expected) Fourier transform radial mean

Comparison

independent random sampling

stratified/jittered sampling

blue-noise sampling
Comparison

stratified/jittered sampling
(4 samples per pixel)

Poisson-disk sampling
(4 samples per pixel)
More in CSE 272!

- reconstruction vs integration
- low-discrepancy sampling
- higher-dimensional sampling
- blue noise + low-discrepancy sampling
- perceptual quality
- ...

...
Textures
Real-world surfaces are colorful!


photo from K.C. Alfred
An option: assign a color to each triangle

- **pros**
  - simple
  - easy to edit (just paint on triangles)

- **cons**
  - couples geometric complexity with color complexity
  - hard to **filter**
    - more on this next lecture
In practice: texture mapping

- assign a "UV" 2D vector to each point on the surface

![Image: texture mapping diagram]

(image from pbrt-v2 pbrt.org/scenes-v2)
UV mapping

• “unwrap” a surface and map it to a 2D square

• automatic UV mapping is an active research area

http://staff.ustc.edu.cn/~fuxm/projects/Peeling/index.html
Distortion is often unavoidable

- ideally we want
  - area-preserving (large areas map to large areas)
  - conformal (angles between any two curves are preserved)
- a theorem from Euler [1775]:
  - for a sphere-to-square projection, a conformal map cannot be area-preserving
  - an area-preserving map cannot be conformal

https://en.wikipedia.org/wiki/Mercator_projection
An area-preserving projection of earth
A few simple “automatic” UV mapping

spherical mapping
cylindrical mapping
planar mapping
Spherical mapping $(\theta, \phi)$
Cylindrical mapping

\((\theta, r)\)
Planar mapping

(i, u, v)
Obtain UV by interpolating values from vertices

\[ uv = (1 - b_1 - b_2)uv_0 + b_1 uv_1 + b_2 uv_2 \]
3D Textures

use a volumetric texture, don’t need UV mapping then

Quiz: what do we use textures for?
Quiz: what do we use textures for?

- color map
- normal map
- roughness map
- displacement map
Textures as BRDF parameters
Procedural textures: textures as programs

```python
def stripe(x, y, z, u, v):
    if int(x) % 2 == 0:
        return red
    else:
        return white
```

1984
Shade Trees
Robert L. Cook

1985
An Image Synthesizer
Ken Perlin
Procedural textures: textures as programs

```python
def ramp(x, y, z, u, v):
    v = (sin(x) + 1) / 2
    return (1 - v) * magenta +
    v * yellow
```

https://www.csie.ntu.edu.tw/~cyy/courses/rendering/16fall/lectures/handouts/chap10_textures.pdf
Procedural textures: textures as programs

```python
def ring(x, y, z, u, v):
    v = (x - center.x)^2 +
        (y - center.y)^2
    if int(v) % 2 == 0:
        return white
    else:
        return red
```

https://www.csie.ntu.edu.tw/~cyy/courses/rendering/16fall/lectures/handouts/chap10_textures.pdf

1984 Shade Trees
Robert L. Cook

1985 An Image Synthesizer
Ken Perlin
Procedural noise

want to represent random natural variation
Procedural noise

want to represent random natural variation
Perlin noise

• a way to procedurally generate stochastic textures

• used in TRON (1982)!
  (first Hollywood film that used 3D shaded graphics)

https://www.csie.ntu.edu.tw/~cyy/courses/rendering/16fall/lectures/handouts/chap10_textures.pdf
Idea: smoothly interpolate white noise
def Noise1D(x):
    xi0 = int(x)
    xi1 = xi0 + 1
    val0, val1 = hash(xi0), hash(xi1)
    t0, t1 = x - xi0, xi1 - x
    w0 = 6 * t0**5 - 15 * t0**4 + 10 * t0**3
    w1 = 6 * t1**5 - 15 * t1**4 + 10 * t1**3
    return w0 * val0 + w1 * val1

see Perlin [2002] for the weight derivation
Perlin’s Noise function

```python
def Noise2D(x):
    xi0, yi0 = int(x), int(y)
    xi1, yi1 = xi0 + 1, yi0 + 1
    val00, val01 = hash(xi0, yi0), hash(xi0, yi1)
    val10, val11 = hash(xi1, yi0), hash(xi1, yi1)
    t0x, t1x = x - xi0, xi1 - x
    t0y, t1y = y - yi0, yi1 - y
    w00 = w(t0x) * w(t0y)
    ...
    return w00 * val00 + ...
```

see Perlin [2002] for the weight derivation
Scale of the inputs determines the smoothness of the noise

\[ \text{Noise2D}(x/n) \]
Combining multiple Noises at different scales

\[ \text{Noise1D}(x) + \frac{1}{2} \times \text{Noise1D}(2x) + \frac{1}{4} \times \text{Noise1D}(4x) + \ldots \]

Combining multiple Noises at different scales
Turbulence

\[(1/2) \cdot \text{Noise3D}(2x) + (1/4) \cdot \text{Noise3D}(4x) + \ldots\]
Marble

perturb uv using turbulence
Map generation

high -> green
middle -> brown
low -> blue

https://medium.com/@yvanscher/playing-with-perlin-noise-generating-realistic-archipelagos-b59f004d8401
Map generation

high -> white
mid-high -> gray
middle -> green
mid-low -> brown
low -> blue

https://medium.com/@yvanscher/playing-with-perlin-noise-generating-realistic-archipelagos-b59f004d8401
Inverse Perlin noise
Adding details to smoke simulation using Perlin’s turbulence noise

(won an Oscar technical achievement award at 2012!)
Next: texture filtering