Texture Mapping II

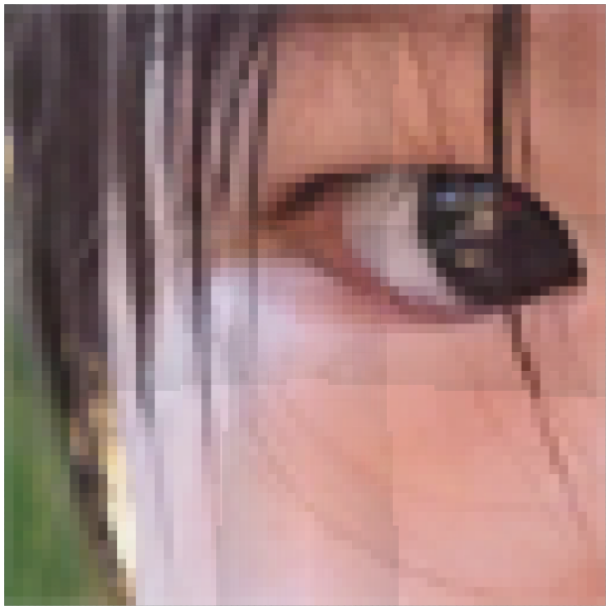
Texture Magnification

(What if the texture is too small?)

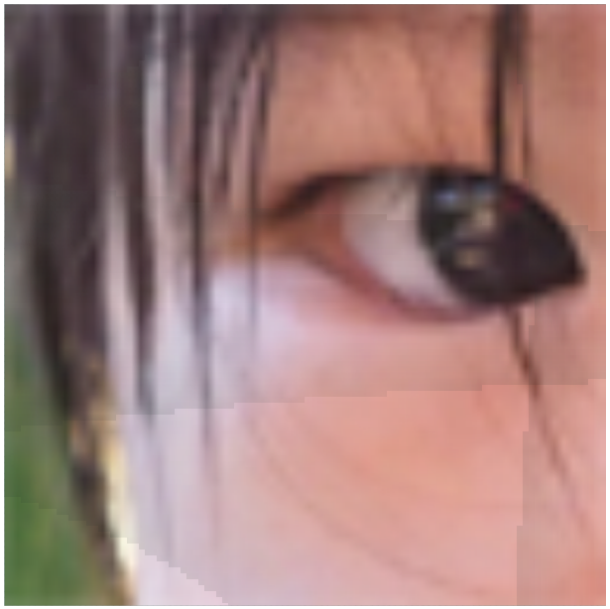
Texture Magnification - Easy Case

Generally don’t want this — insufficient texture resolution

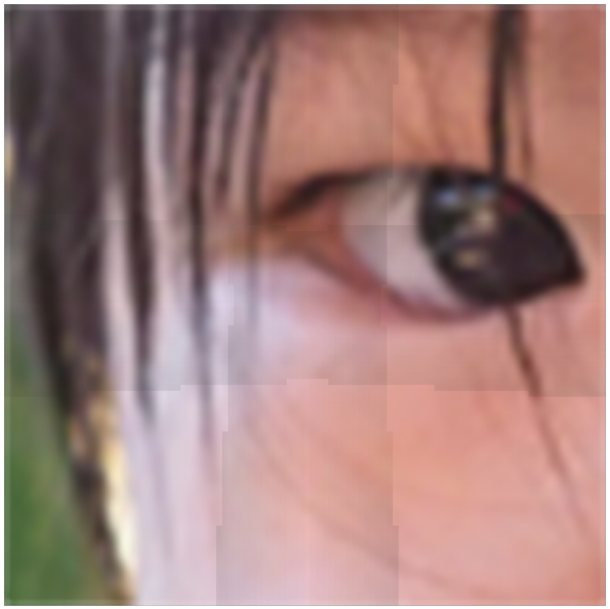
A pixel on a texture — a **texel**



Nearest

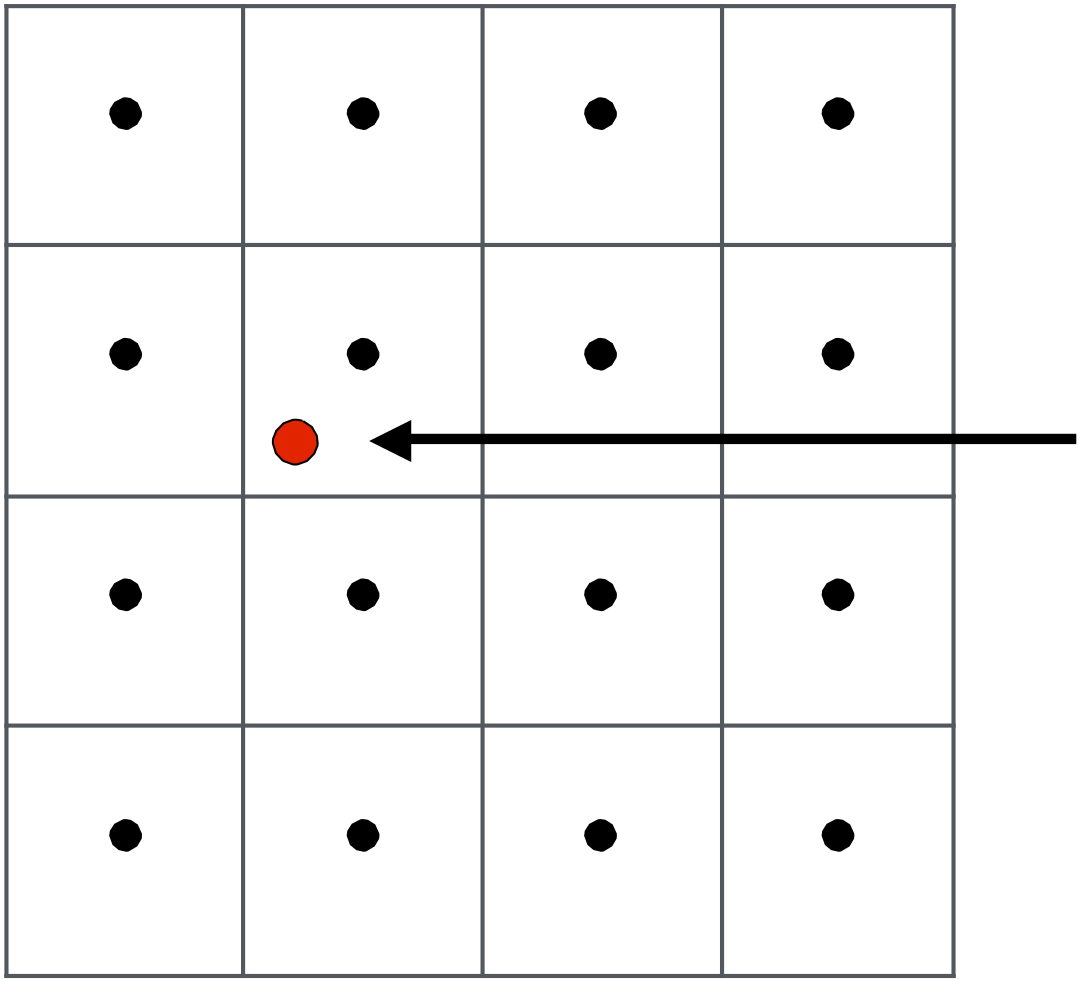


Bilinear



Bicubic

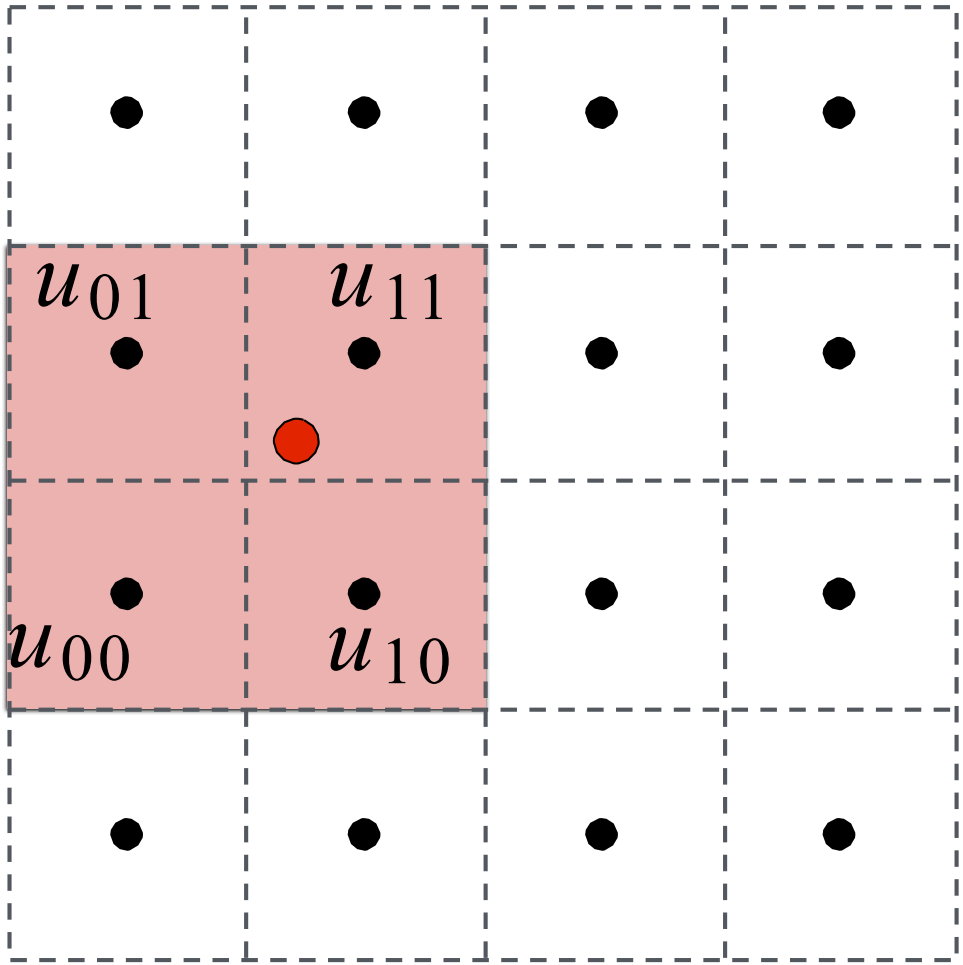
Bilinear Interpolation



Want to sample texture value f(x,y) at red point

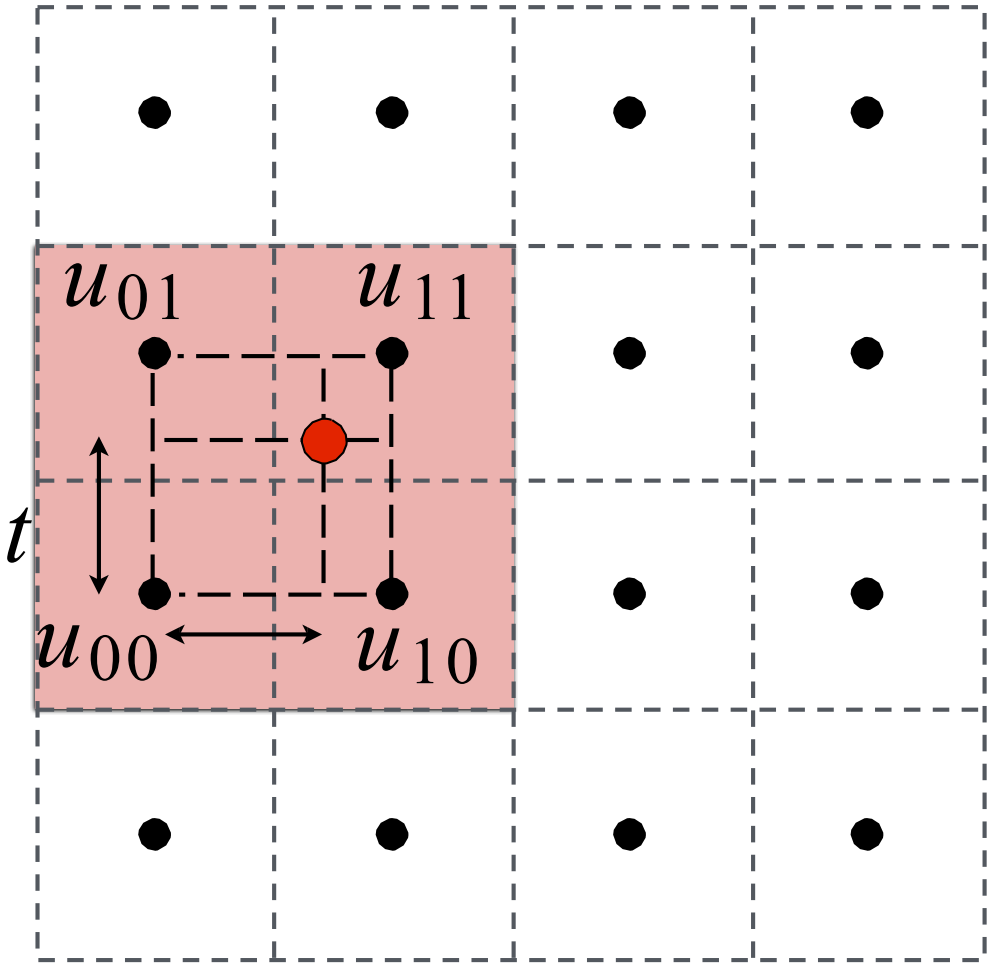
Black points indicate texture sample locations

Bilinear Interpolation



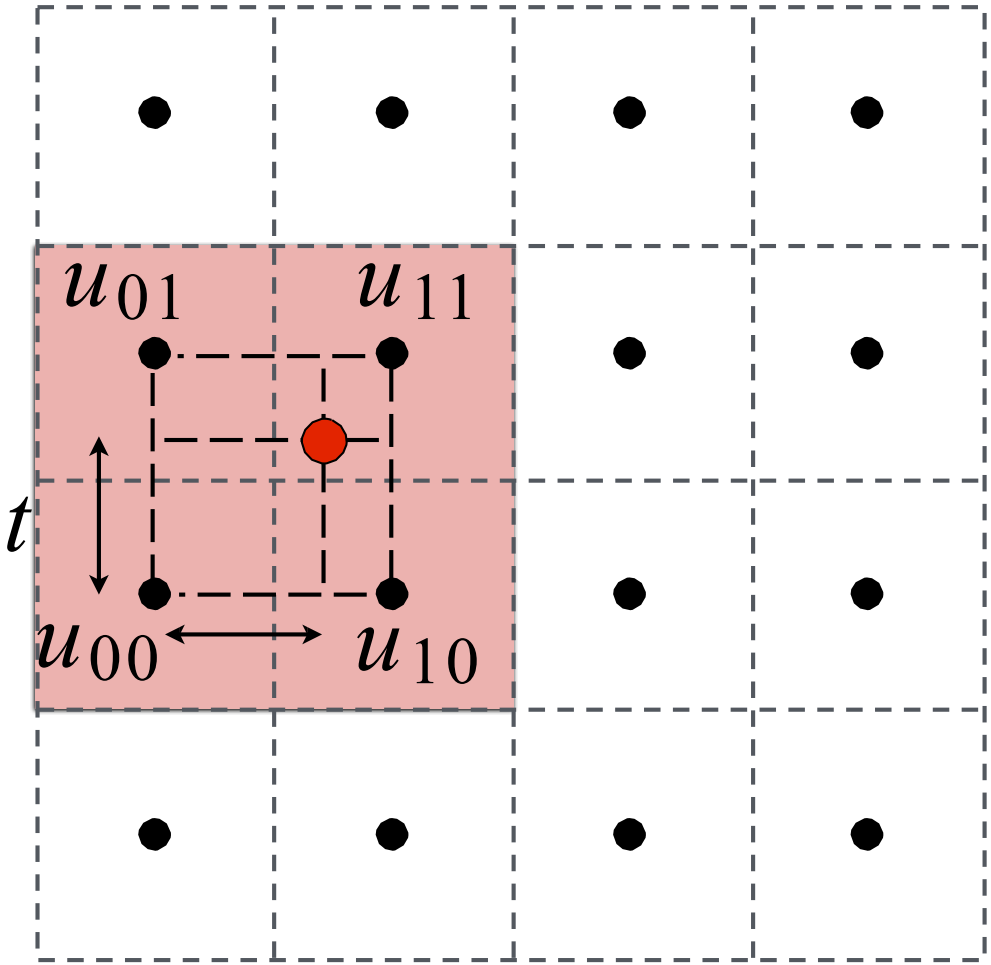
Take 4 nearest sample locations, with texture values as labeled.

Bilinear Interpolation



And fractional offsets, (s,t) as shown

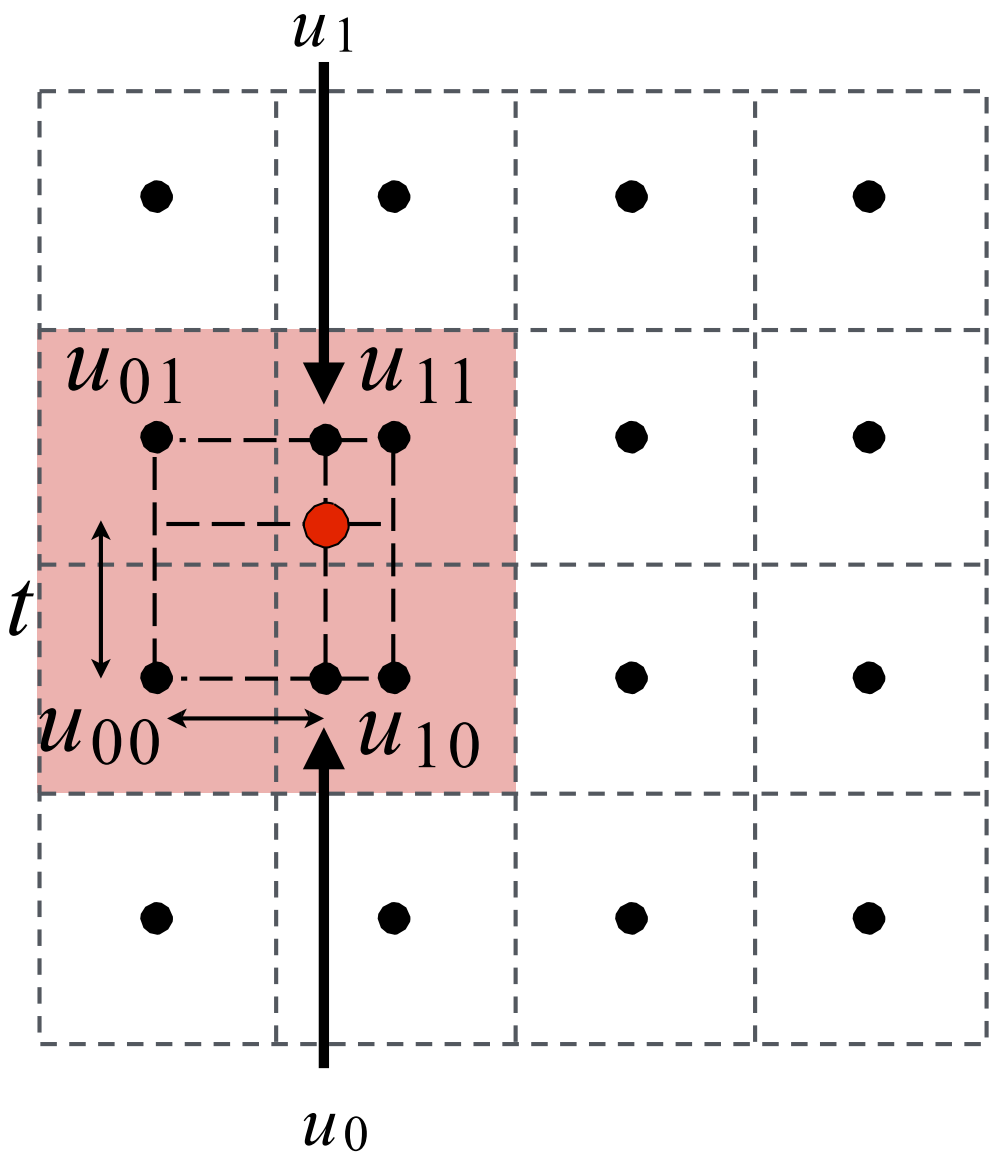
Bilinear Interpolation



Linear interpolation (1D)

$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

Bilinear Interpolation



Linear interpolation (1D)

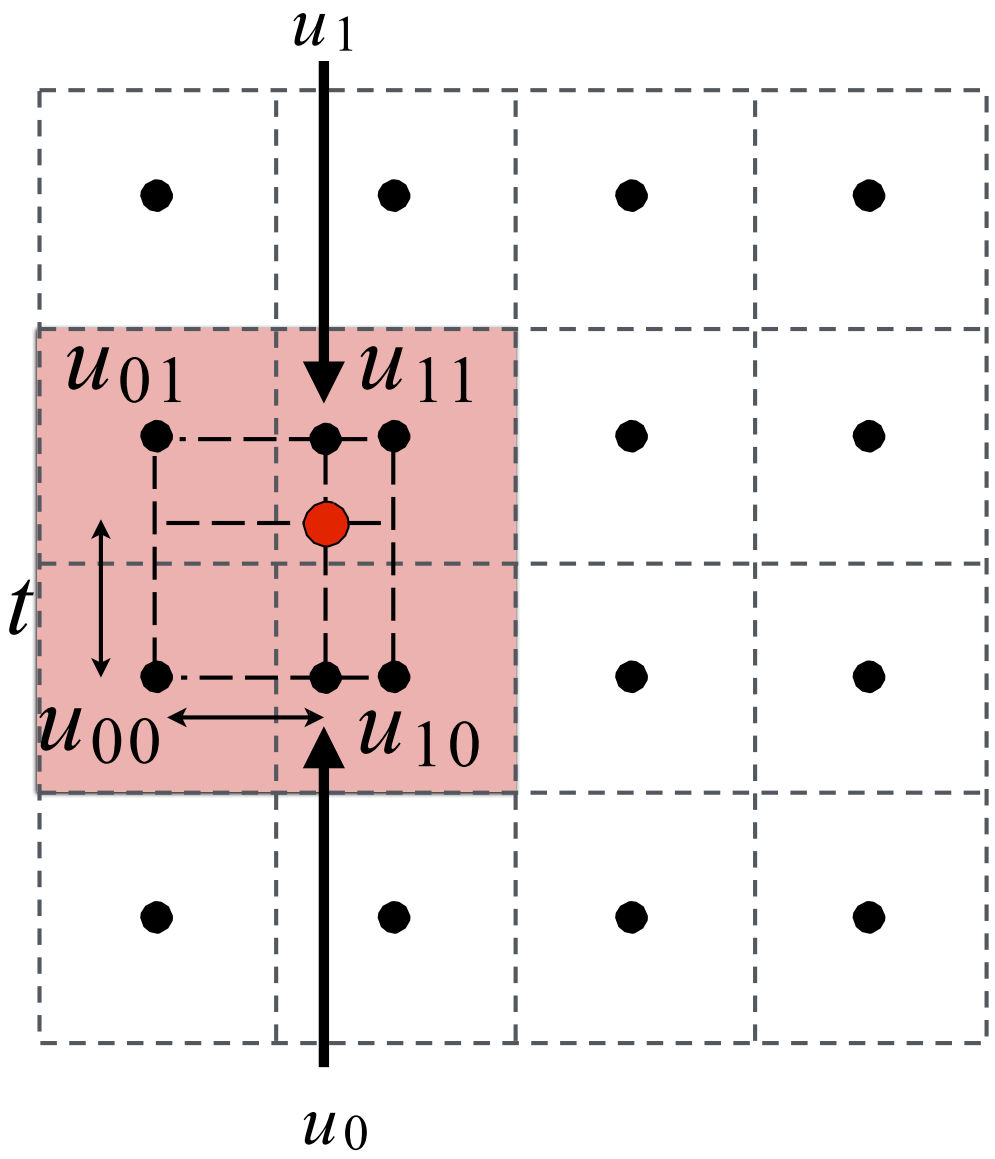
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

Two helper lerps (horizontal)

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

Bilinear Interpolation



Linear interpolation (1D)

$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

Two helper lerps

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

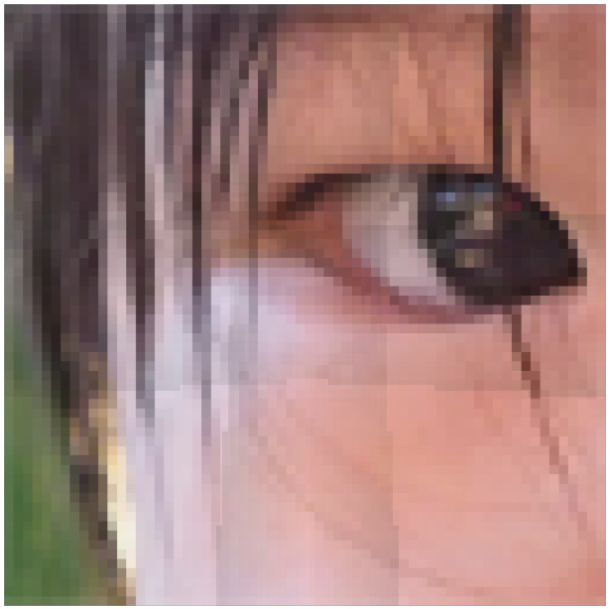
$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

Final vertical lerp, to get result:

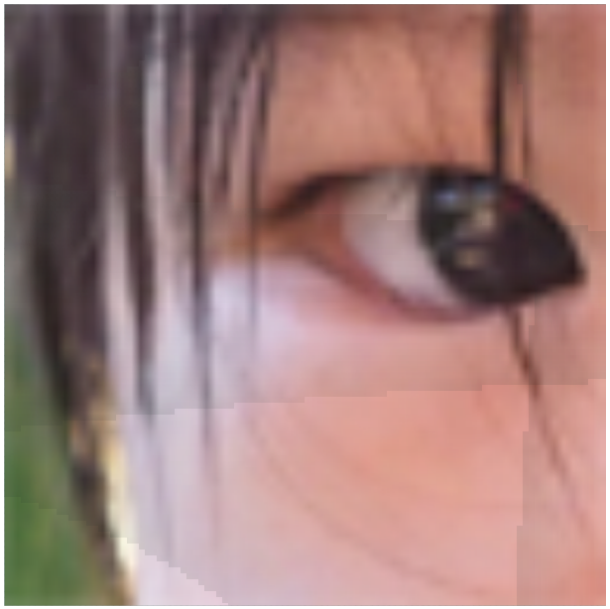
$$f(x, y) = \text{lerp}(t, u_0, u_1)$$

Texture Magnification - Easy Case

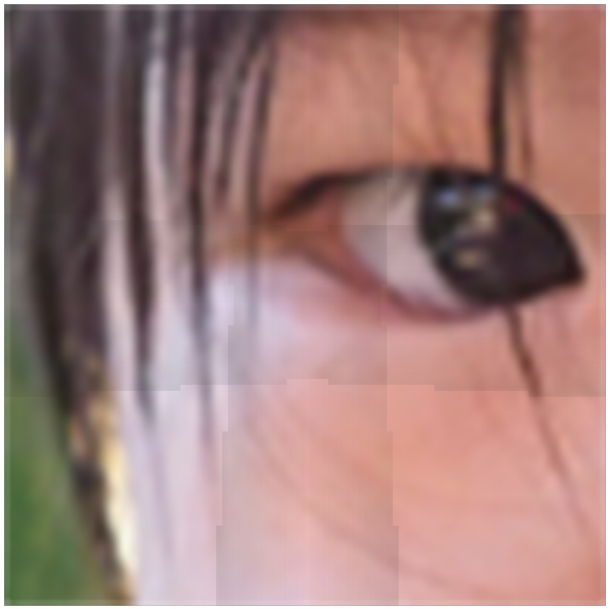
Bilinear interpolation usually gives pretty good results at reasonable costs



Nearest



Bilinear

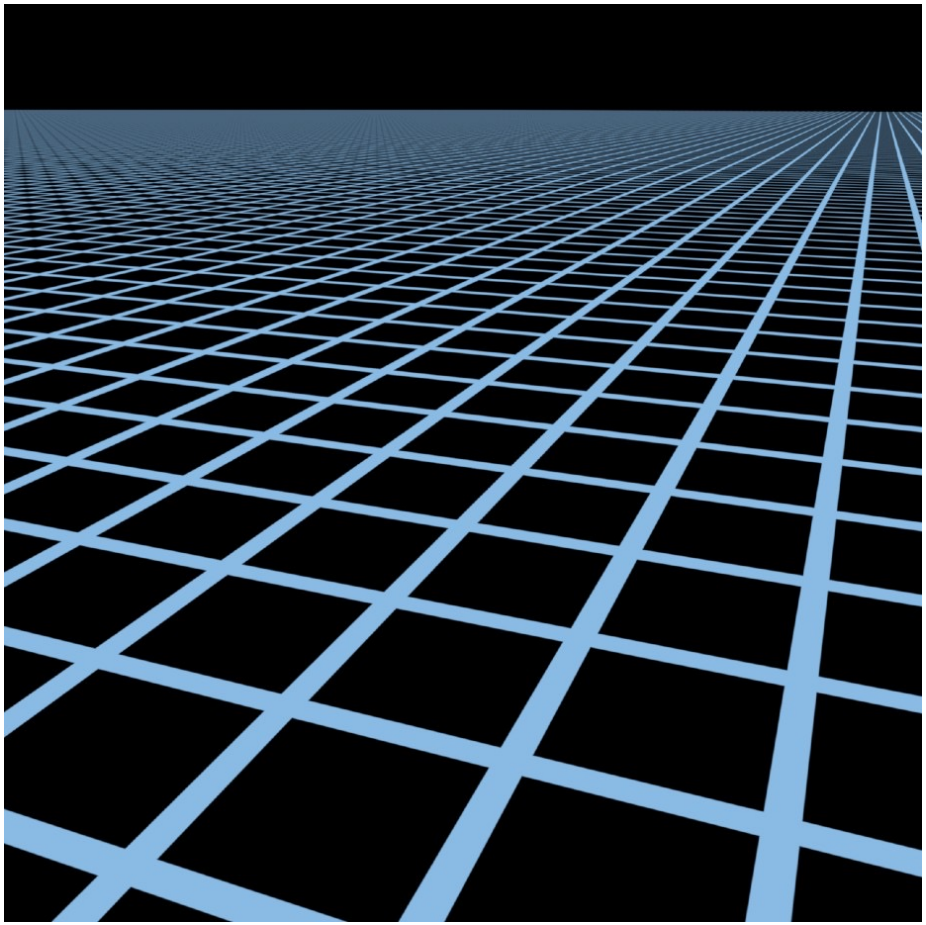


Bicubic

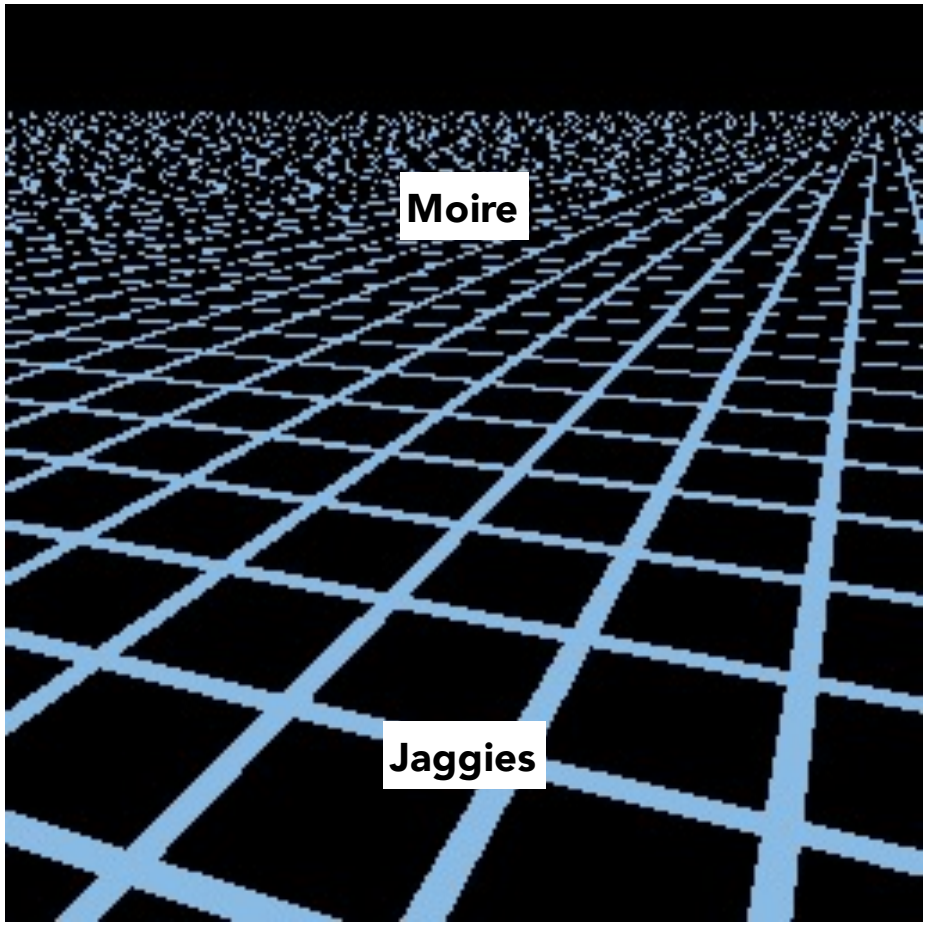
Texture Magnification **(hard case)**

(What if the texture is too large?)

Point Sampling Textures — Problem

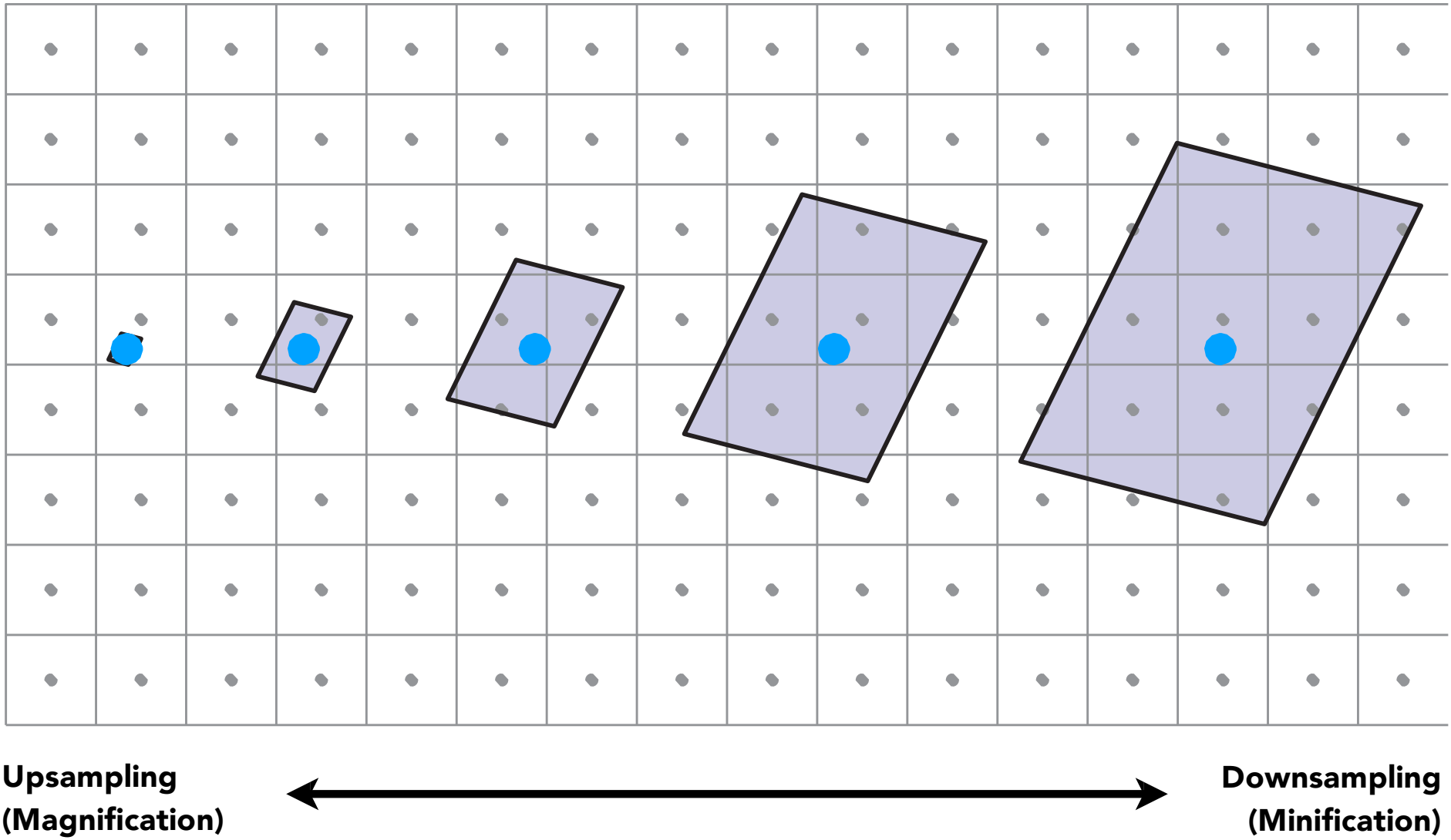


Reference

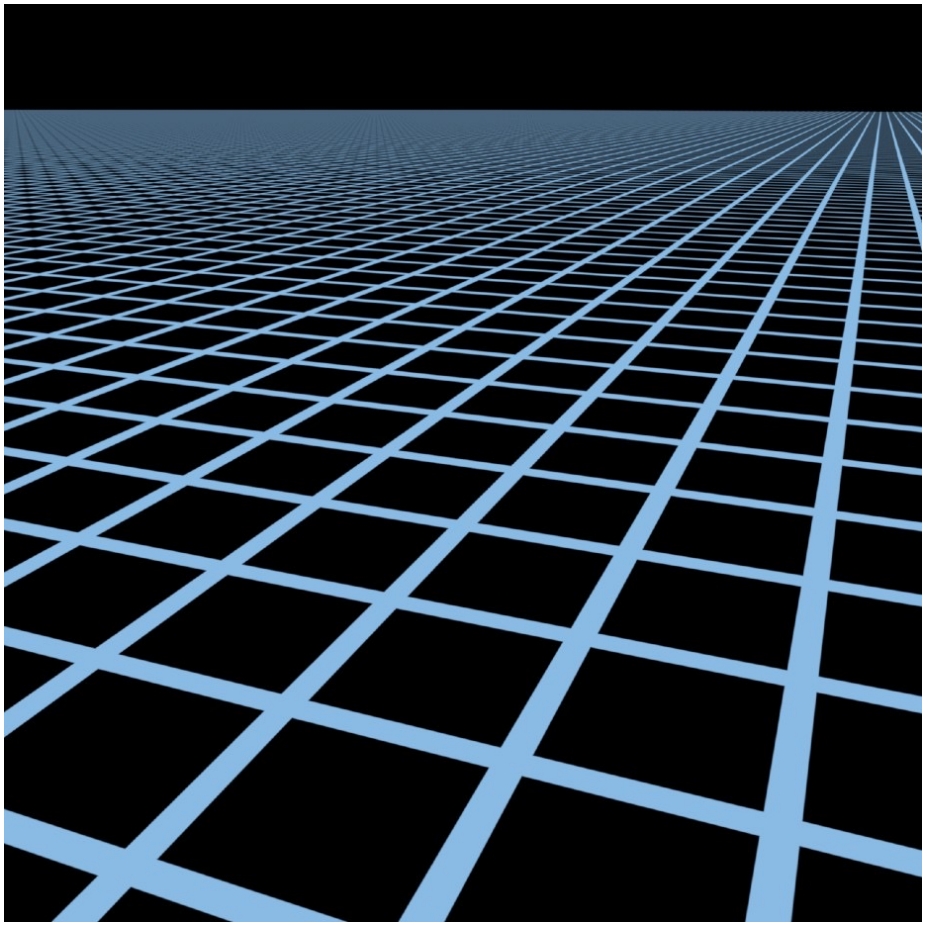


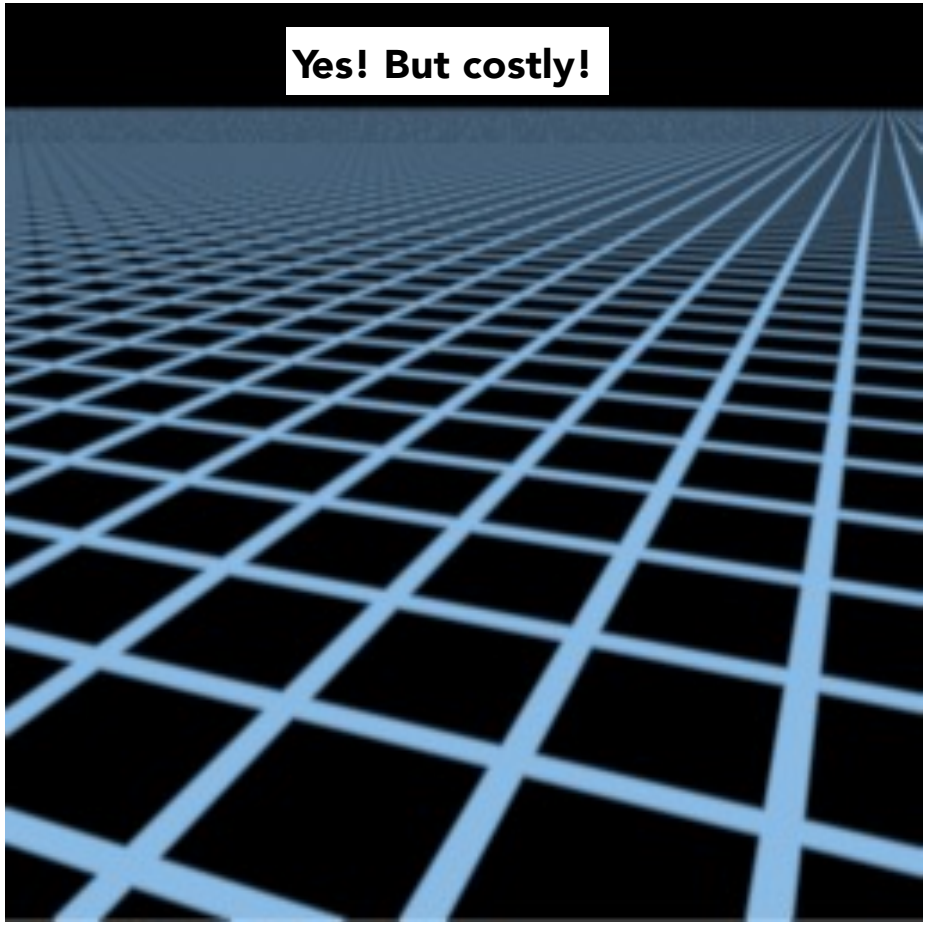
Point sampled

Screen Pixel "Footprint" in Texture



Will Supersampling Do Antialiasing?





512x supersampling

Antialiasing — Supersampling

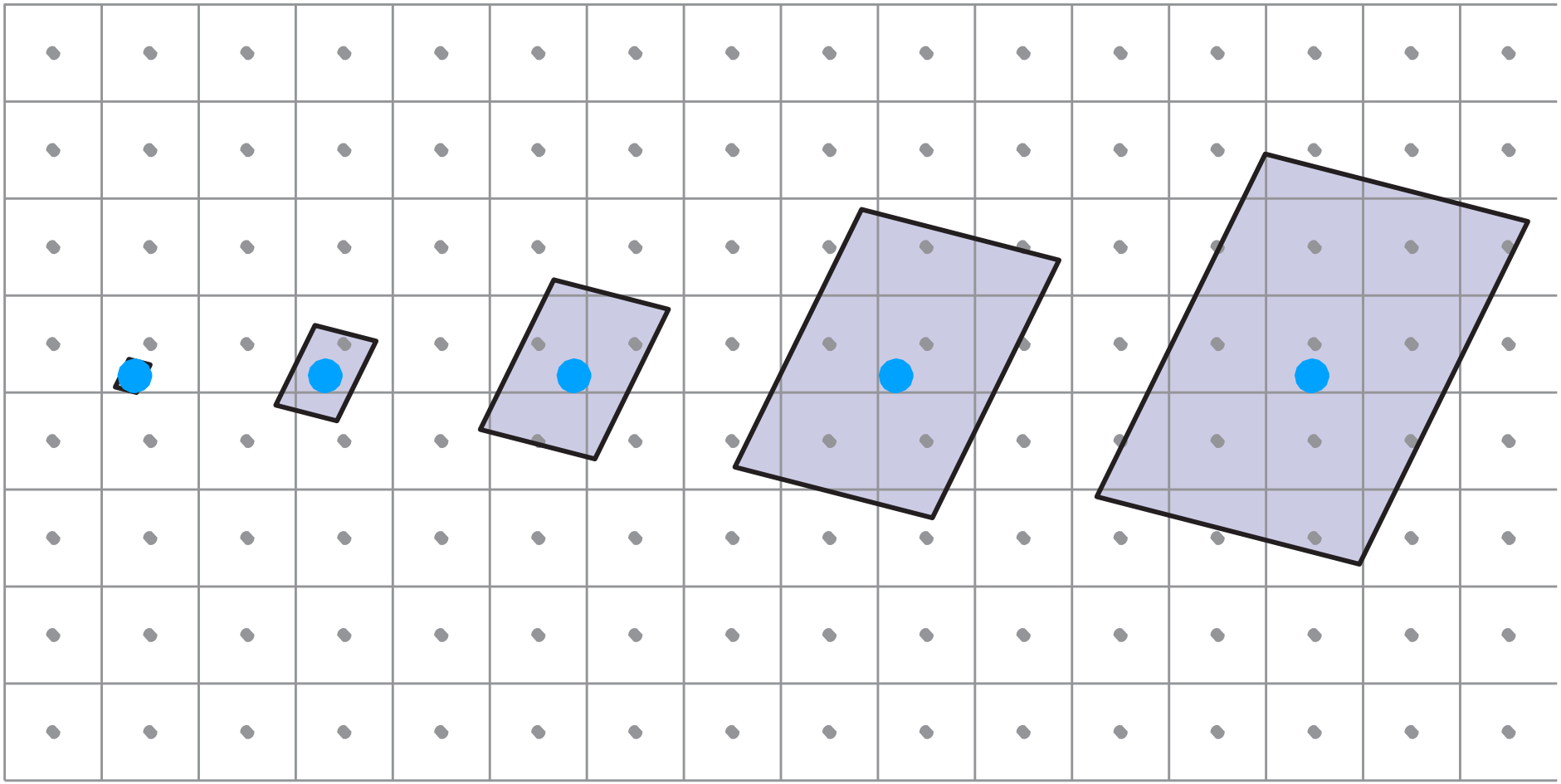
Will supersampling work?

- Yes, high quality, but costly
- When highly minified, many texels in pixel footprint
- Signal frequency too large in a pixel
- Need even higher sampling frequency

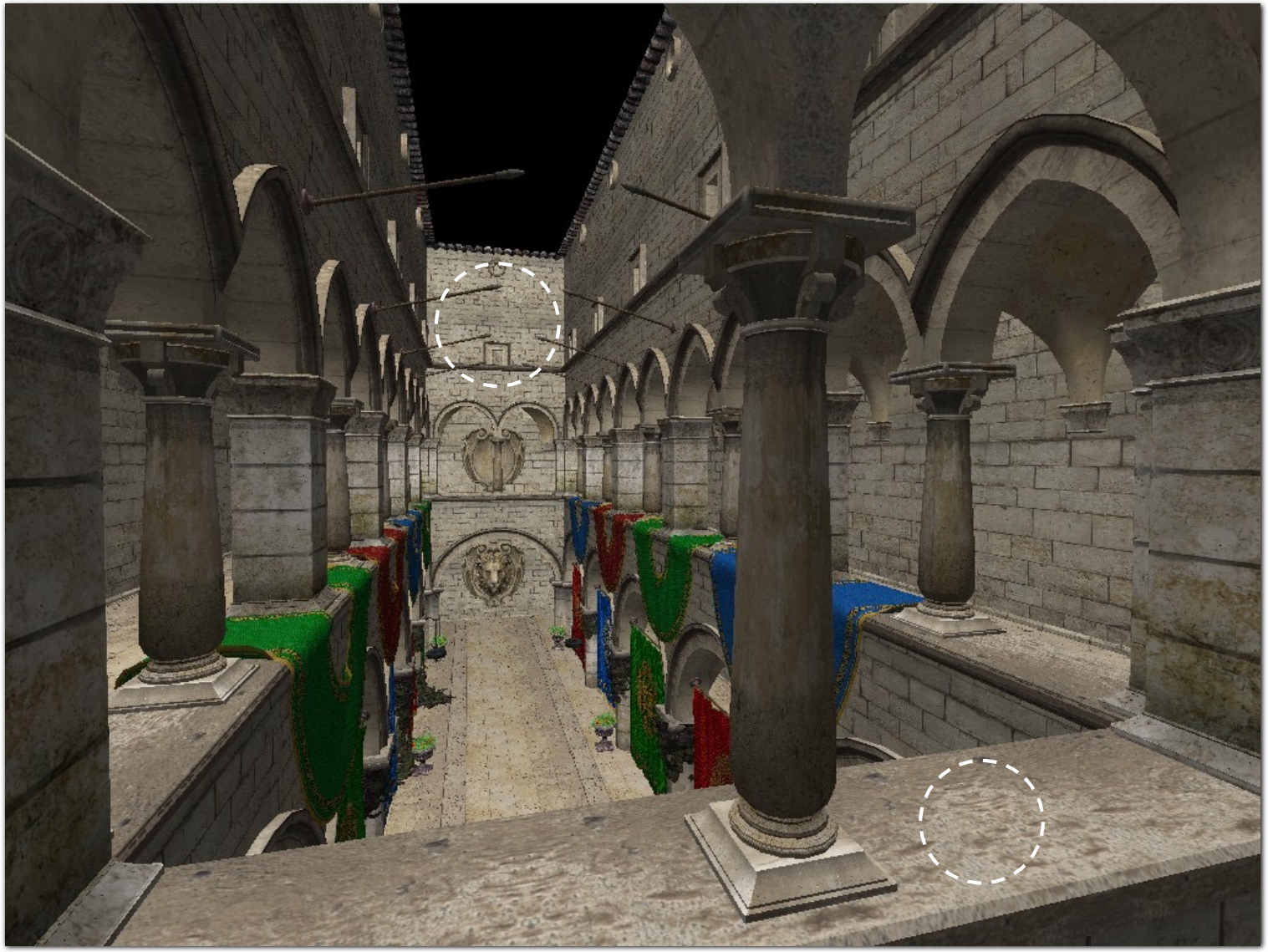
Let’s understand this problem in another way

- What if we don’t sample?
- Just need to **get the average value within a range**!

Point Query vs. (Avg.) Range Query



Different Pixels -> Different-Sized Footprints



Mipmap

Allowing (fast, approx., square) range queries

Mipmap (L. Williams 83)

“Mip” comes from the Latin “multum in parvo", meaning a multitude in a small space

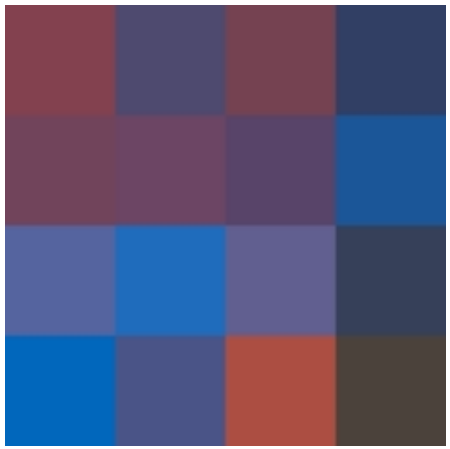
Level 0 = 128x128

Level 1 = 64x64

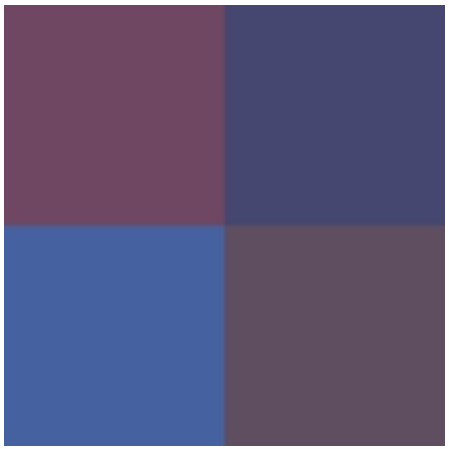
Level 2 = 32x32

Level 3 = 16x16

Level 4 = 8x8



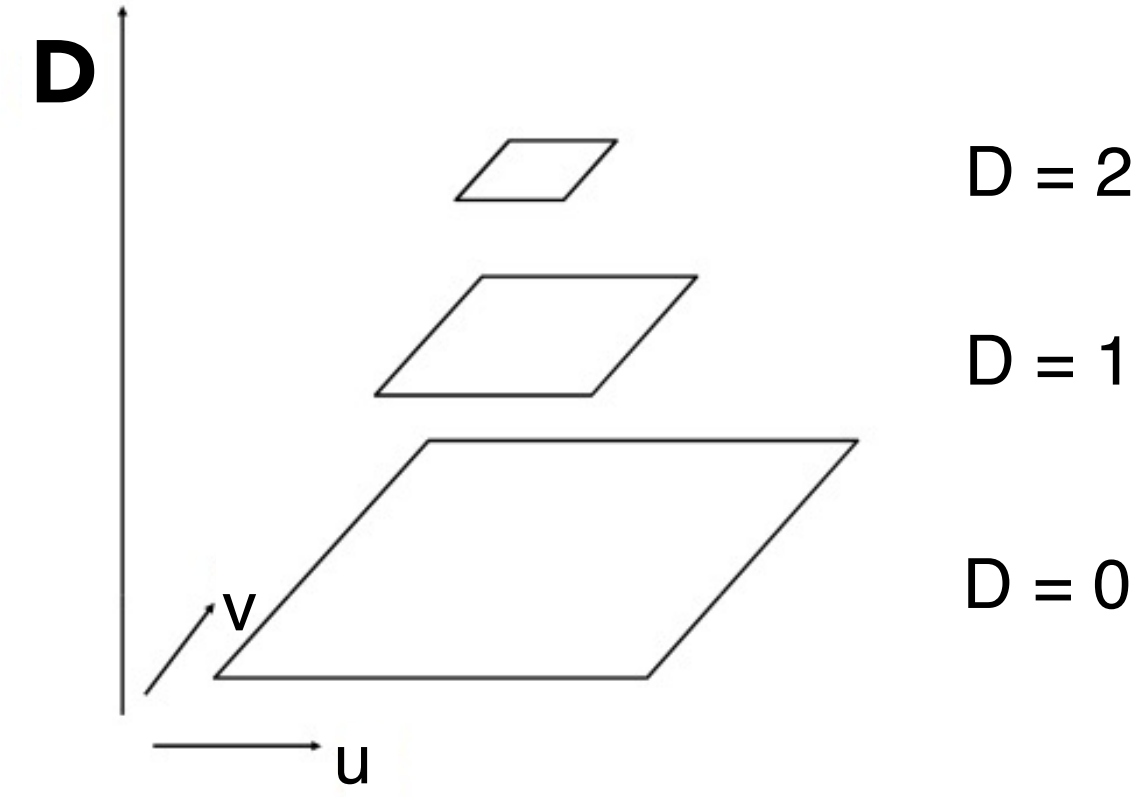
Level 5 = 4x4



Level 6 = 2x2

Level 7 = 1x1

Mipmap (L. Williams 83)

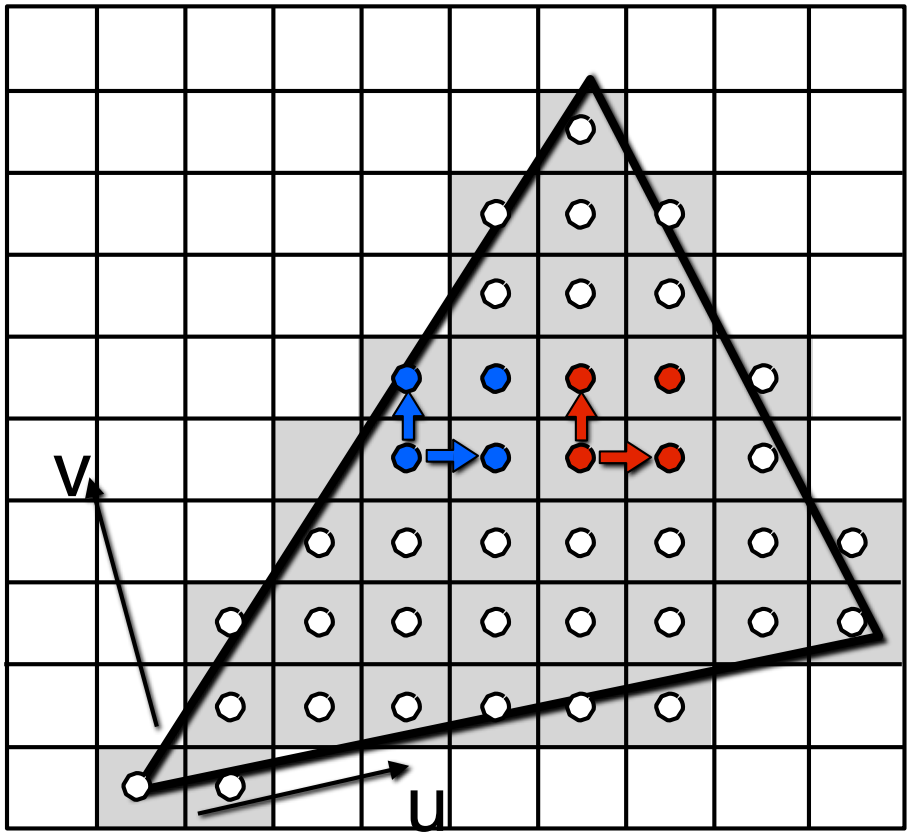


"Mip hierarchy"

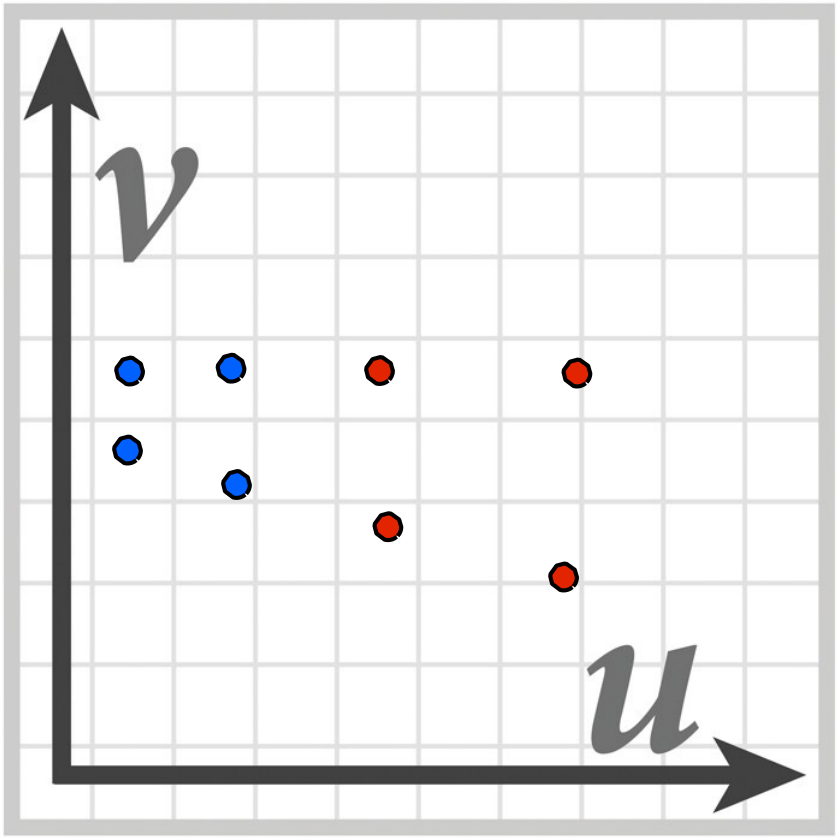
level = D

What is the storage overhead of a mipmap?

Computing Mipmap Level D



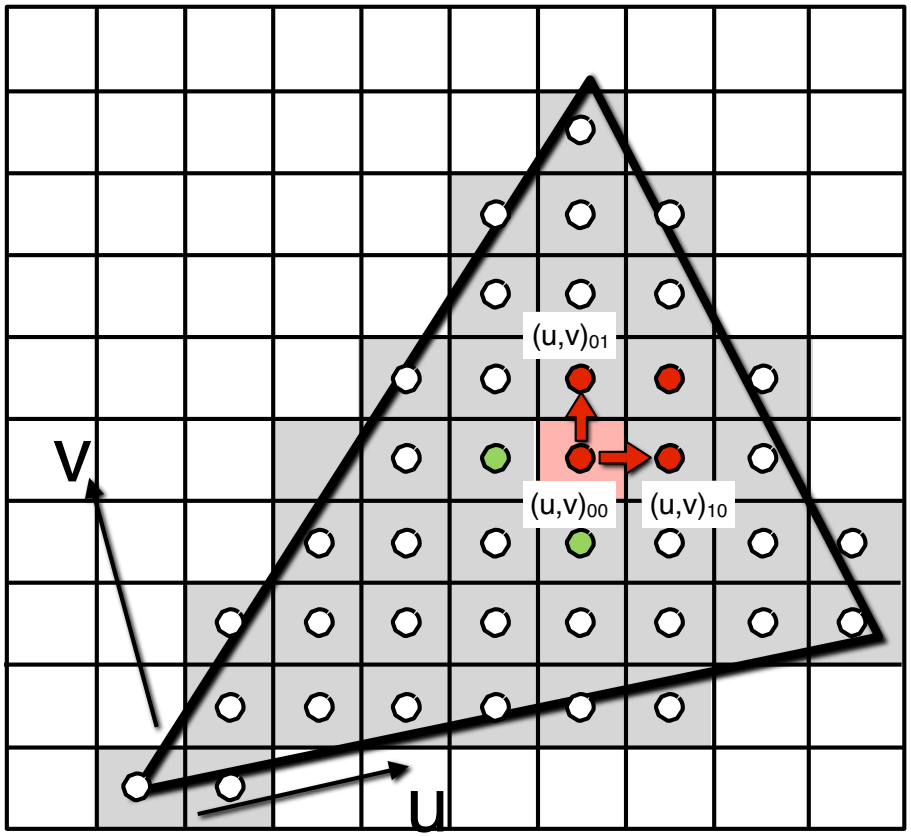
Screen space (x,y)

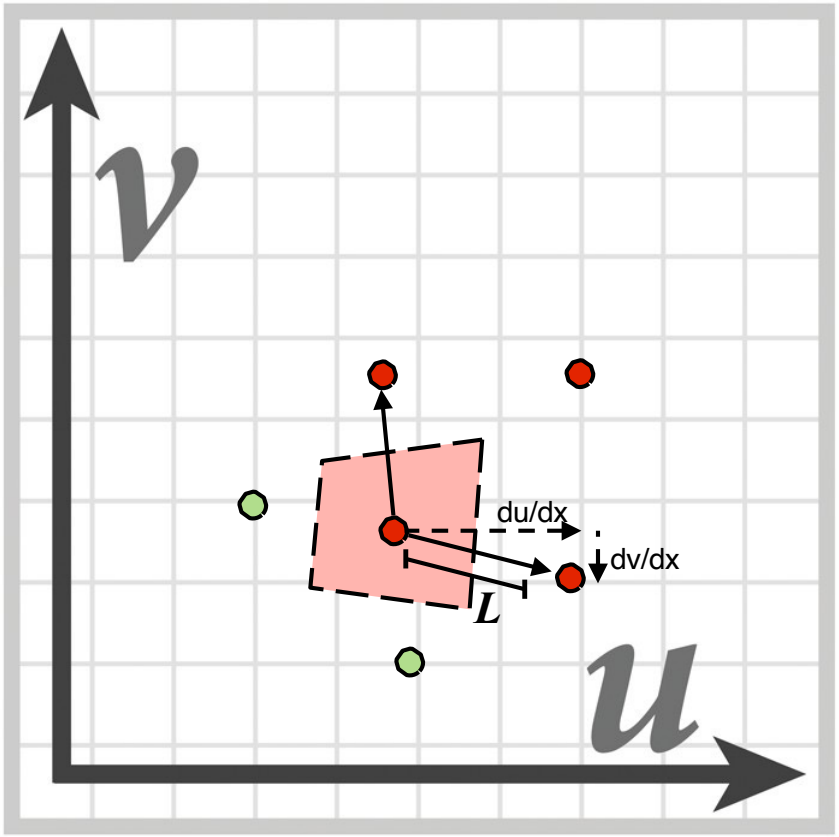


Texture space (u,v)

Estimate texture footprint using **texture coordinates of neighboring screen samples**

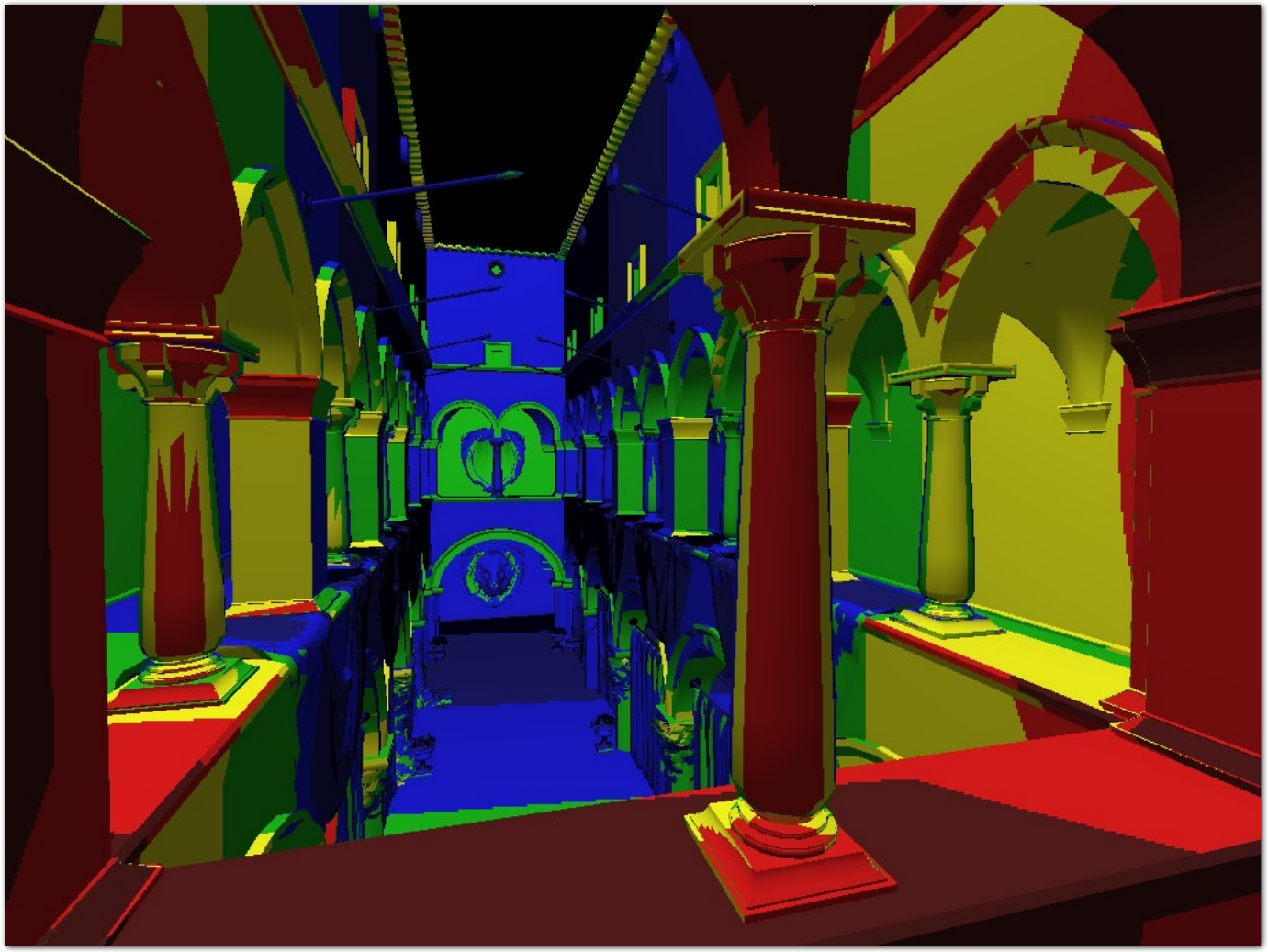
Computing Mipmap Level D





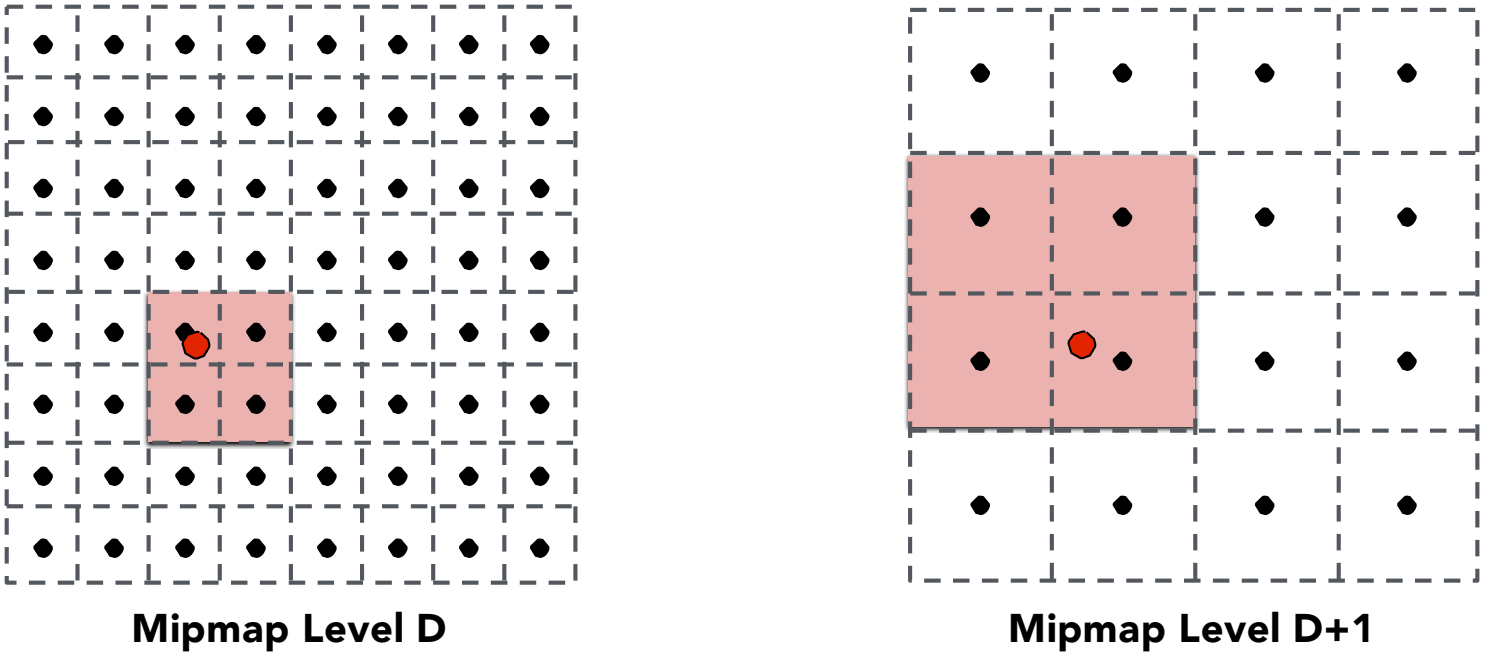
$$D = \log_2 L \qquad L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Visualization of Mipmap Level

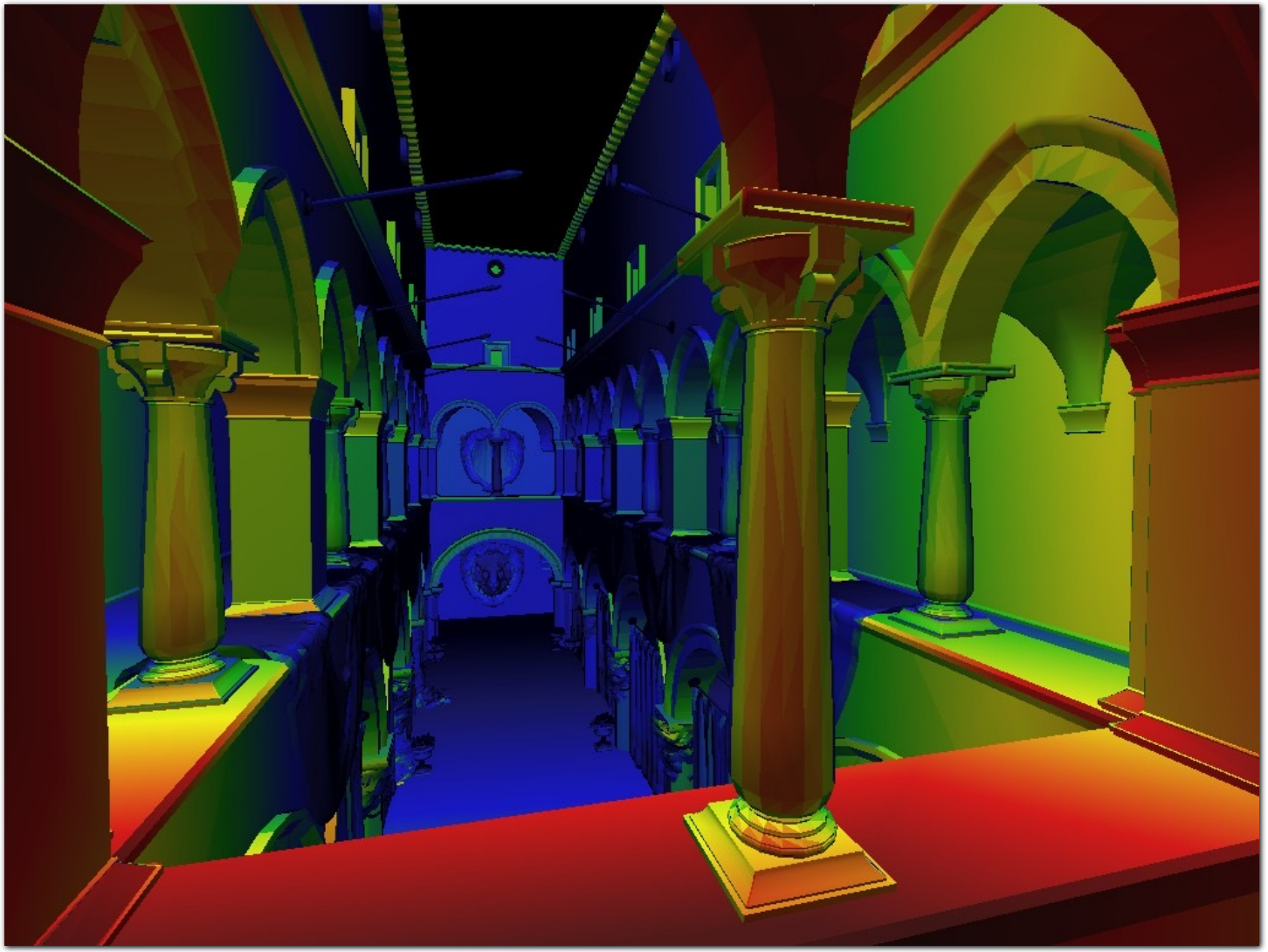


D rounded to nearest integer level

Trilinear Interpolation

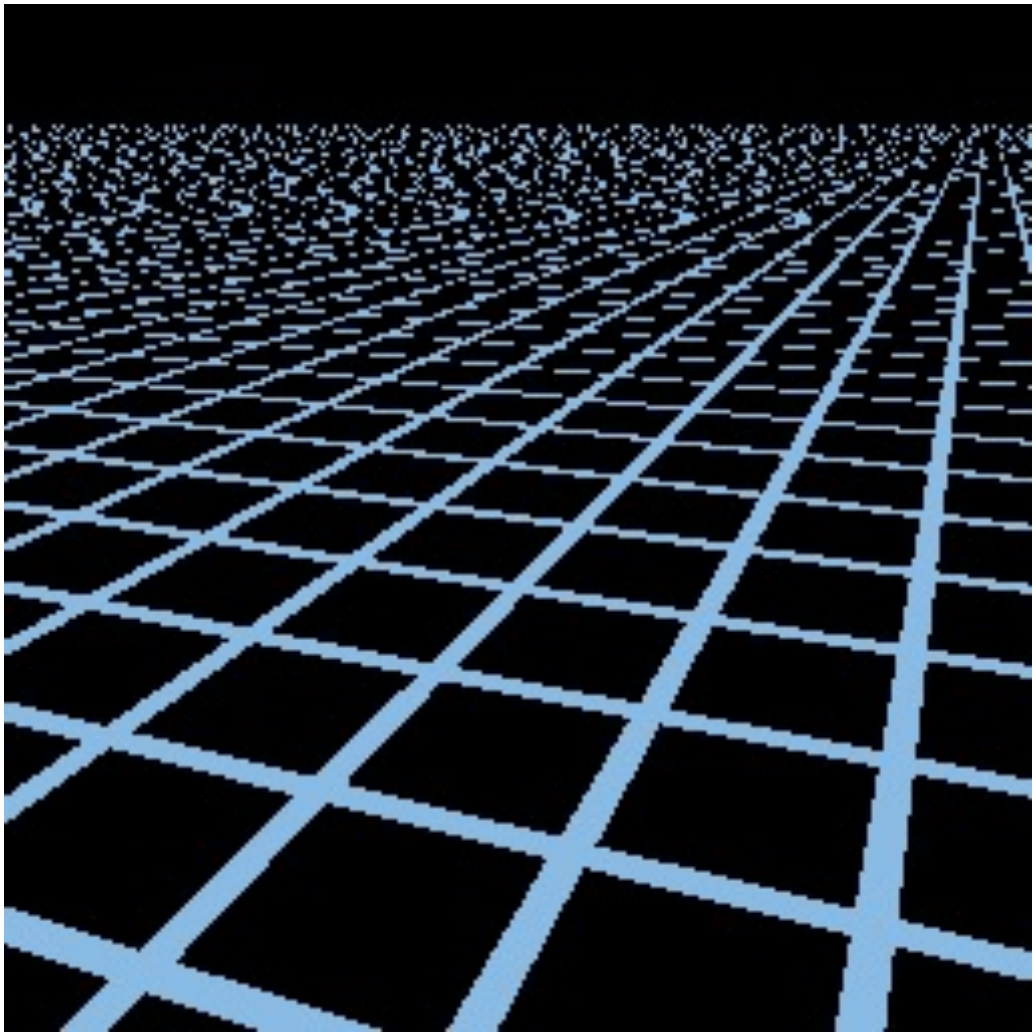


Visualization of Mipmap Level



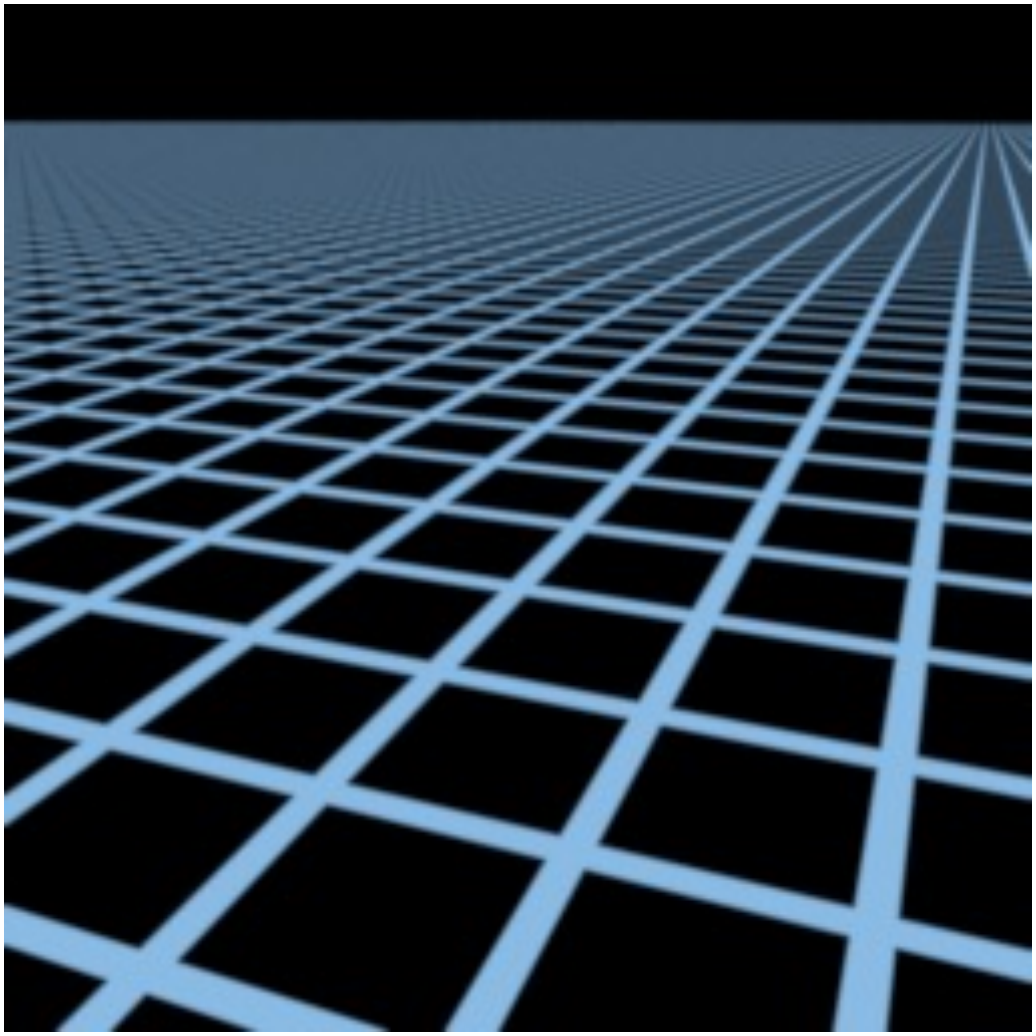
Trilinear filtering: visualization of continuous D

Mipmap Limitations



Point sampling

Mipmap Limitations

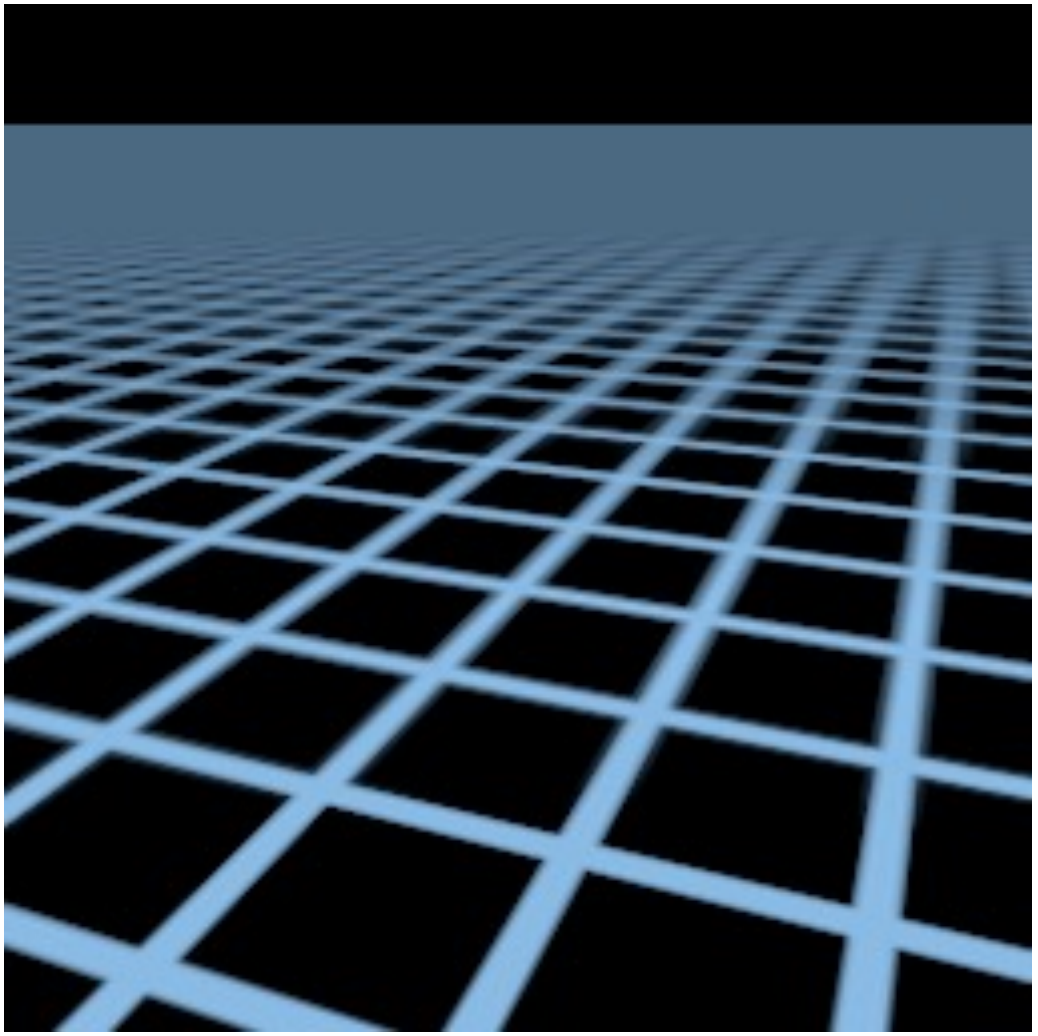


Supersampling 512x (assume this is correct)

Mipmap Limitations

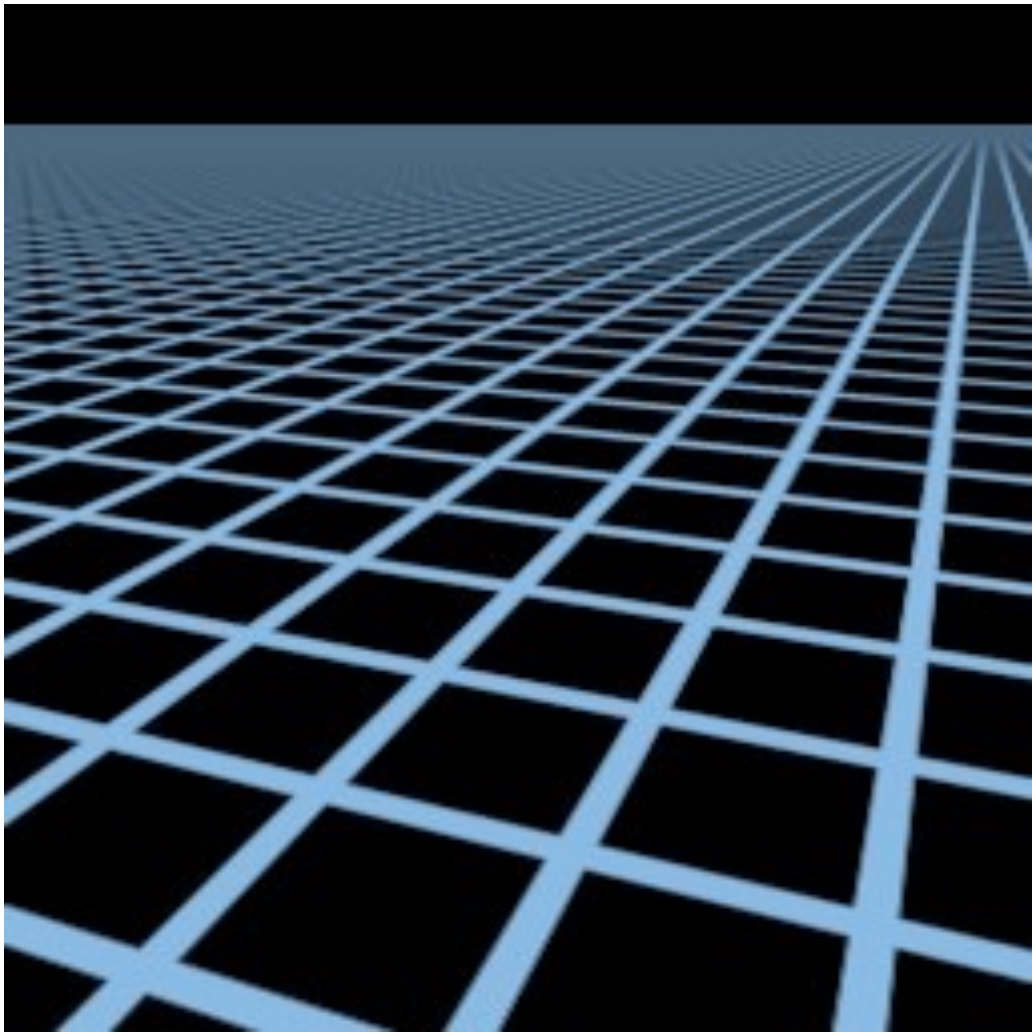
Overblur

Why?



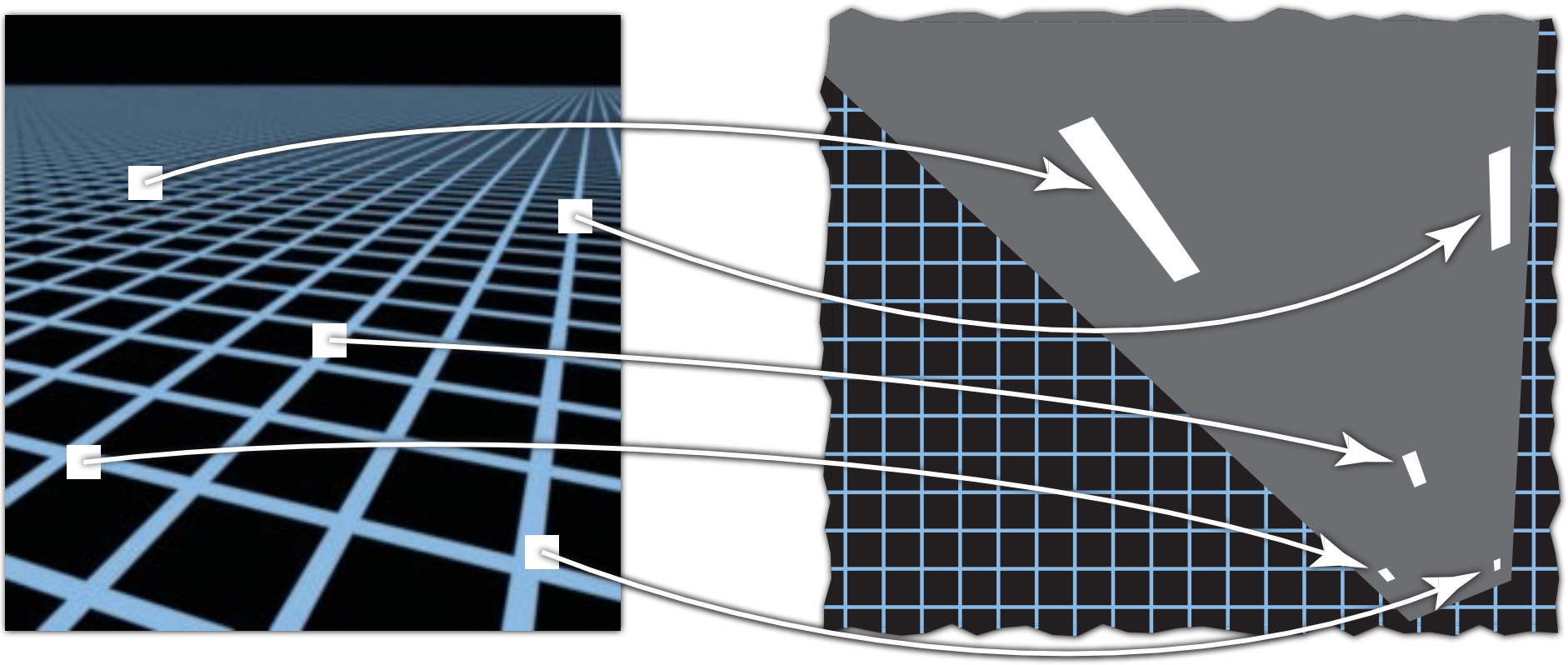
Mipmap trilinear sampling

Anisotropic Filtering



Better than Mipmap!

Irregular Pixel Footprint in Texture



Screen space

Texture space

Anisotropic Filtering

Ripmaps and summed area tables

- Can look up **axis-aligned rectangular zones**
- Diagonal footprints still a problem

Wikipedia

Anisotropic Filtering

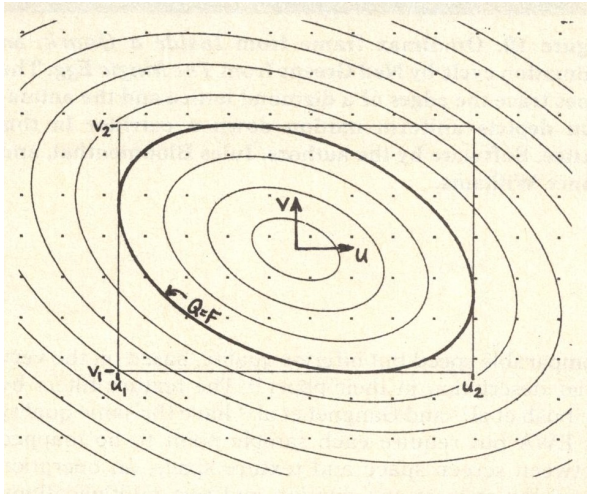
Ripmaps and summed area tables

- Can look up axis-aligned rectangular zones
- Diagonal footprints still a problem

Wikipedia

EWA filtering

- Use multiple lookups
- Weighted average
- Mipmap hierarchy still helps
- Can handle irregular footprints



Greene & Heckbert ‘86

Many, Many Uses for Texturing

In modern GPUs, texture = memory + range query (filtering)

- General method to bring data to fragment calculations

Many applications

- Environment lighting
- Store microgeometry
- Procedural textures
- Solid modeling
- Volume rendering
- …

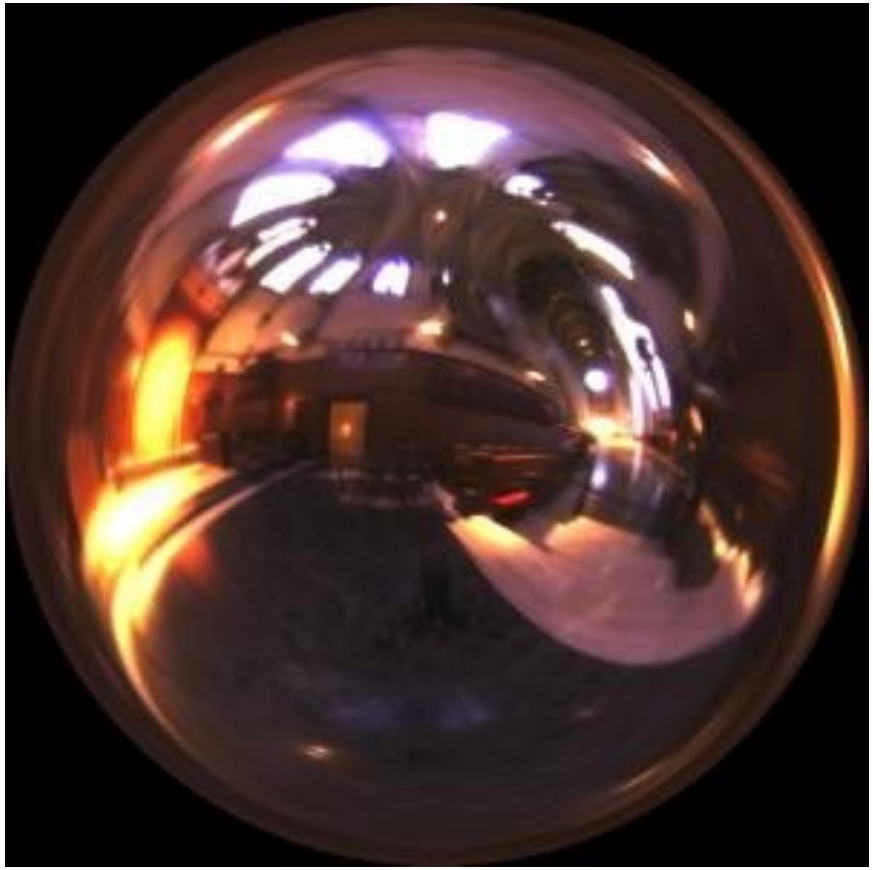
Environment Map

Light from the environment

Rendering with the environment

[Blinn & Newell 1976]

Environmental Lighting



Environment map (left) used to render realistic lighting

Spherical Environment Map

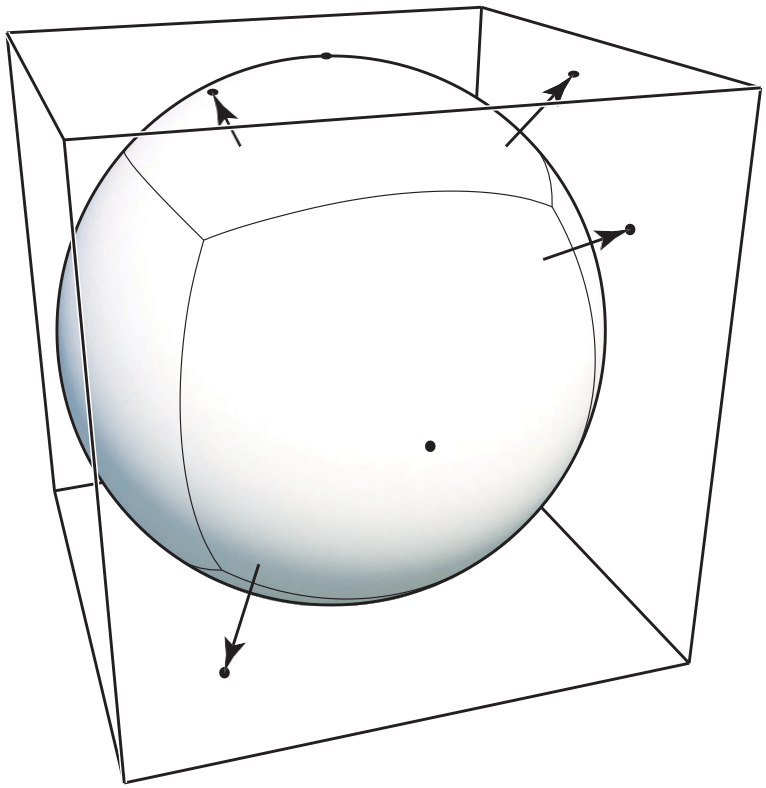
Hand with Reflecting Sphere. M. C. Escher, 1935. lithograph

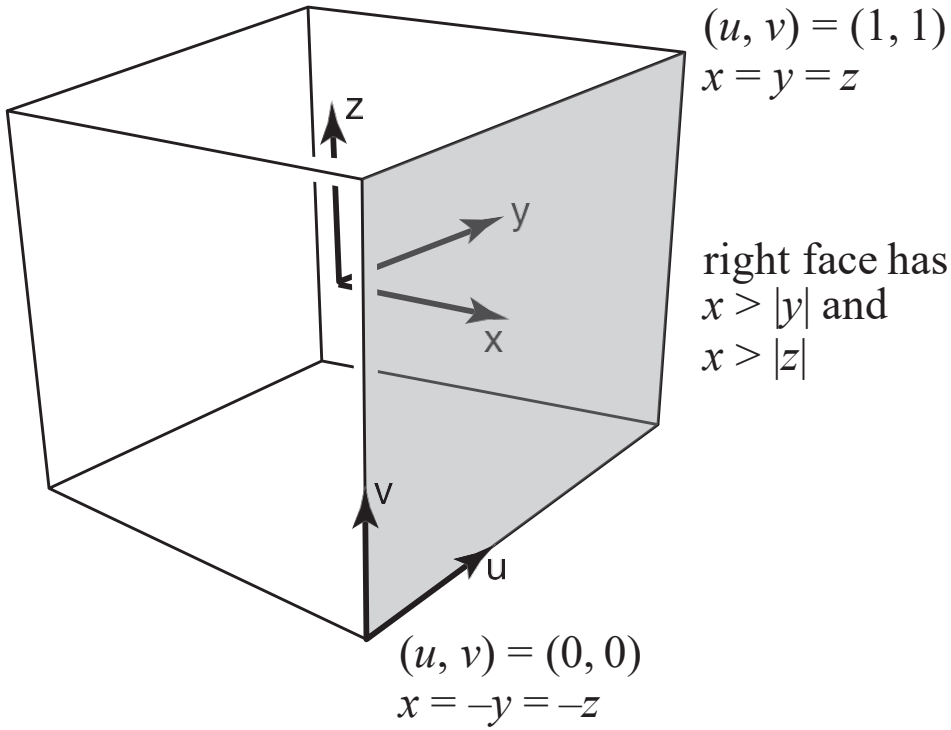
Light Probes, Paul Debevec

Spherical Map — Problem

Prone to distortion (top and bottom parts)!

Cube Map





A vector maps to cube point along that direction.

The cube is textured with 6 square texture maps.

[Emil Persson]

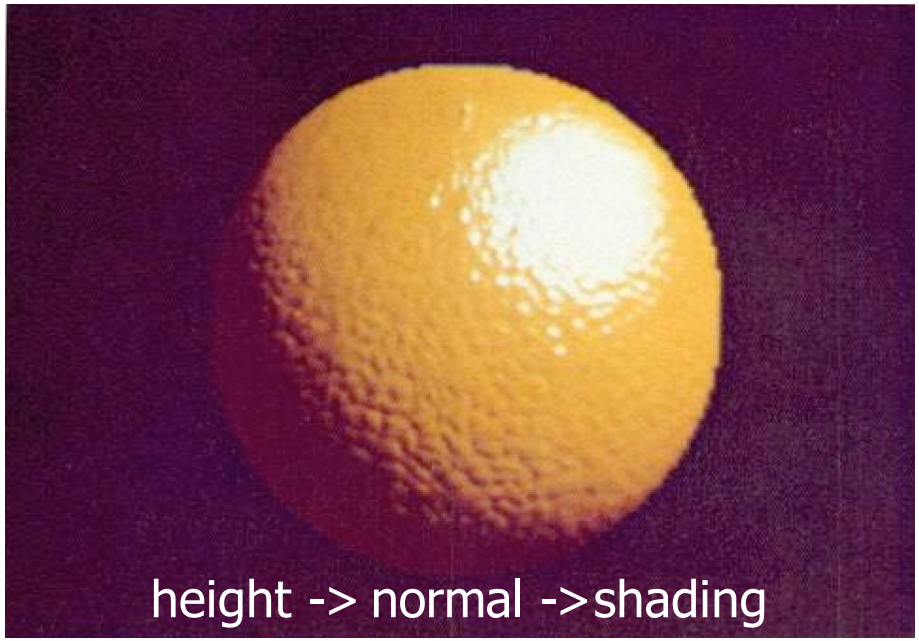
Much less distortion!

Need dir->face computation

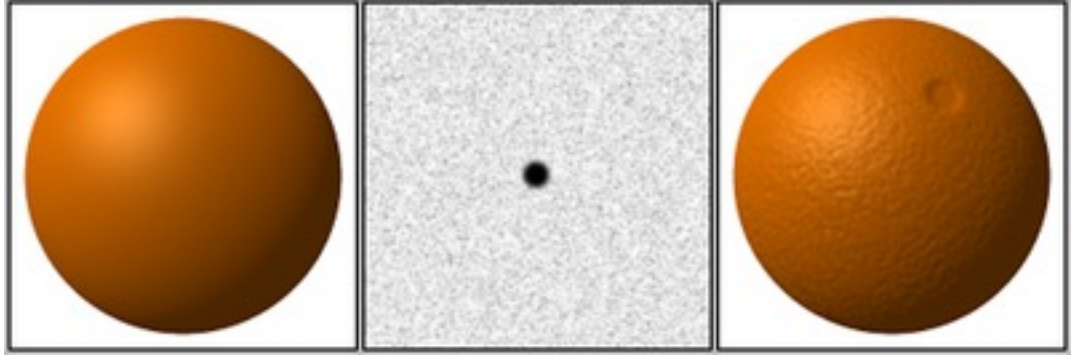
Textures can affect shading!

- Textures doesn't have to only represent colors
  - What if it stores the height / normal?
  - Bump / normal mapping
  - **Fake** the detailed geometry

Relative height to the underlying surface



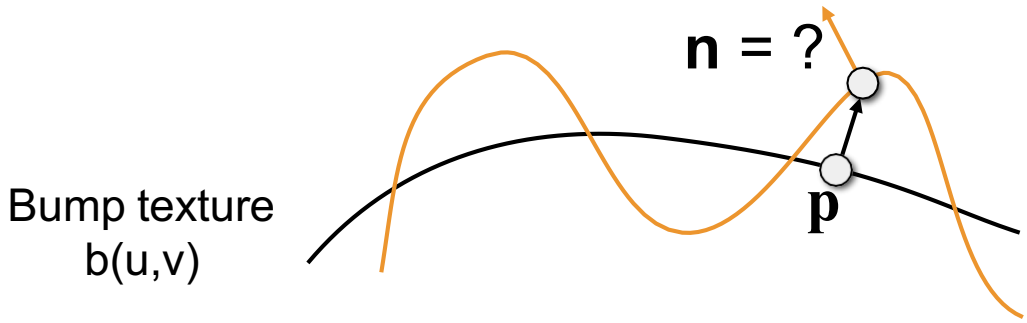
Bump Mapping



wikipedia

Adding surface detail without adding more triangles

- Perturb surface normal per pixel
  (for shading computations only)
- "Height shift" per texel defined by a texture
- How to modify normal vector?



How to perturb the normal

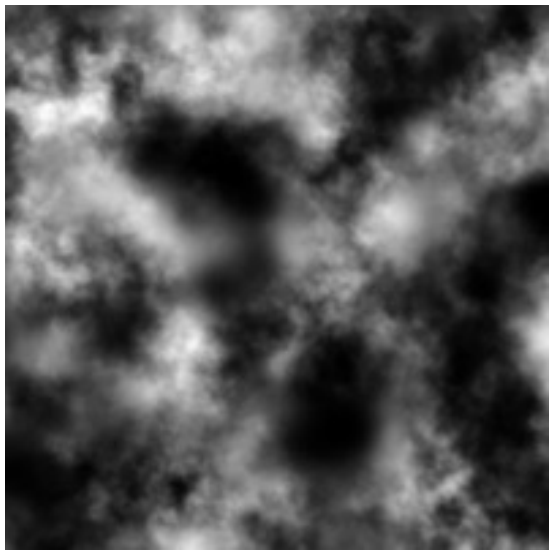
Before a lighting calculation is performed for each visible point (or pixel) on the object's surface:

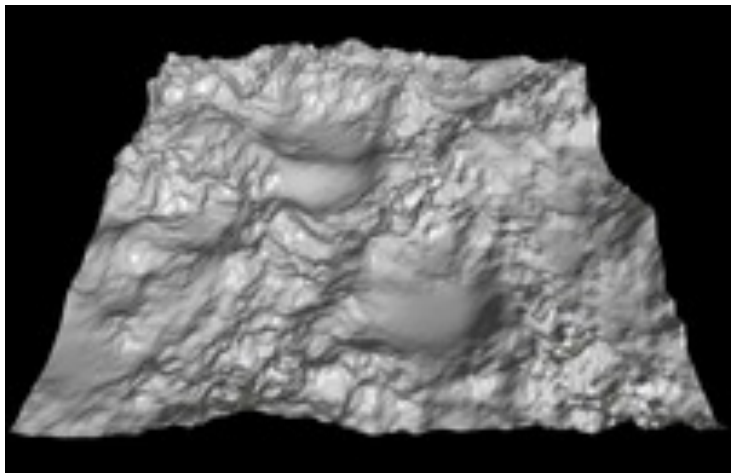
1.Look up the height in the heightmap that corresponds to the position on the surface.

2.Calculate the surface normal of the heightmap, typically using the finite difference method.

3.Combine the surface normal from step two with the true ("geometric") surface normal so that the combined normal points in a new direction.

4. Calculate the interaction of the new "bumpy" surface with lights in the scene using, for example, the Phong reflection model.





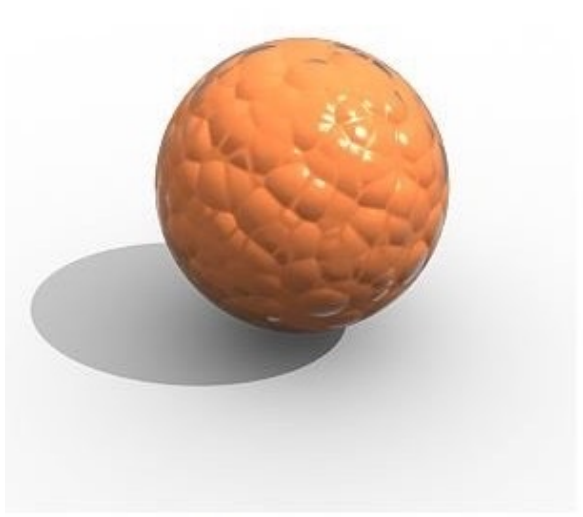
wikipedia

How to perturb the normal (in 3D)

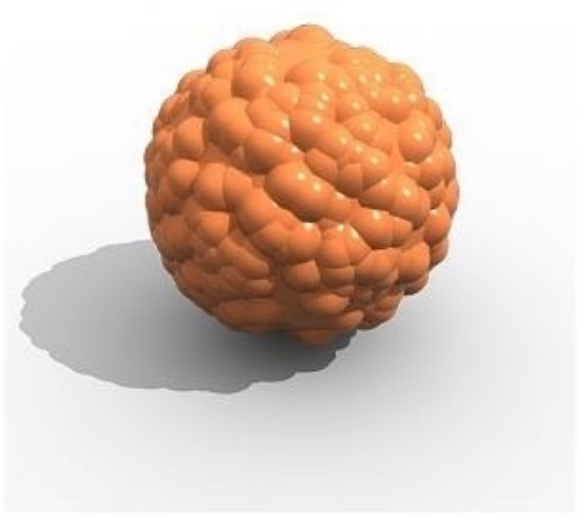
- Original surface normal n(p) = (0, 0, 1)
- Derivatives at p are
  - dp/du = c1 * [h(**u**+1) - h(**u**)]
  - dp/dv = c2 * [h(**v**+1) - h(**v**)]
- Perturbed normal is n = **(-dp/du, -dp/dv, 1)**.normalized()
- Note that this is in **local coordinate**!

Textures can affect shading!

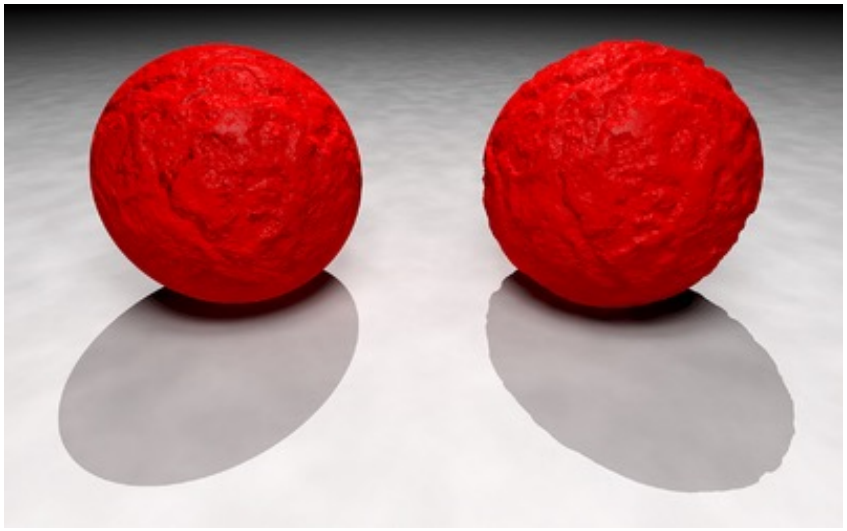
- **Displacement mapping** — a more advanced approach
  - Uses the same texture as in bumping mapping
  - Actually **moves the vertices**



Bump / **Normal** mapping

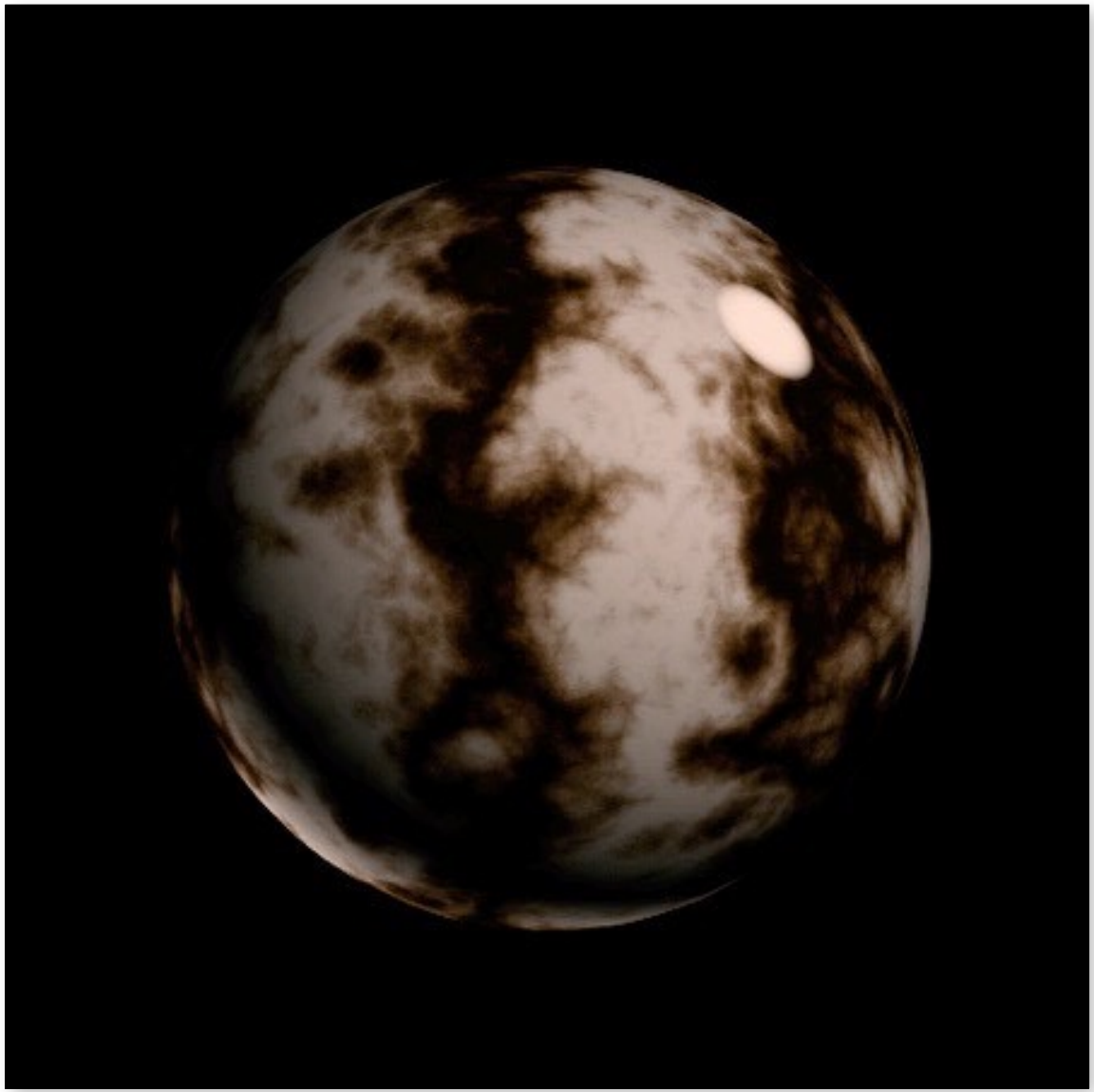


Displacement mapping



wikipedia

3D Procedural Noise + Solid Modeling



Perlin noise, Ken Perlin

Provide Precomputed Shading

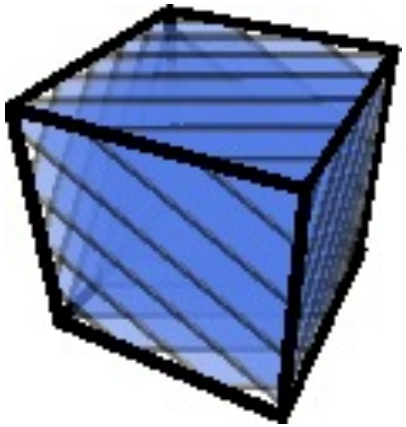
Simple shading

Ambient occlusion texture map

With ambient occlusion

Autodesk

3D Textures and Volume Rendering



Marc Levoy

Thank you!

(And thank Prof. Lingqi Yan, Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)