Adaptive Texture Maps

Martin Kraus and Thomas Ertl
VIS Group, Universität Stuttgart

Introduction

Two Remarks:

- Our goal is to implement “adaptive” texture maps with off-the-shelf graphics hardware.
- This research is about an application of programmable graphics hardware, not a suggestion for new graphics hardware.

Introduction

Two Problems of Hardware-Assisted Texture Mapping:

- Texture data must be specified on uniform grids.
  - We restrict ourselves to the mapping from texture coordinates to texture data.
- Limited texture memory:
  - Usually enough for 2d texture data,
  - Hardly enough for 3d texture data,
  - Usually not enough for 4d texture data.

Introduction

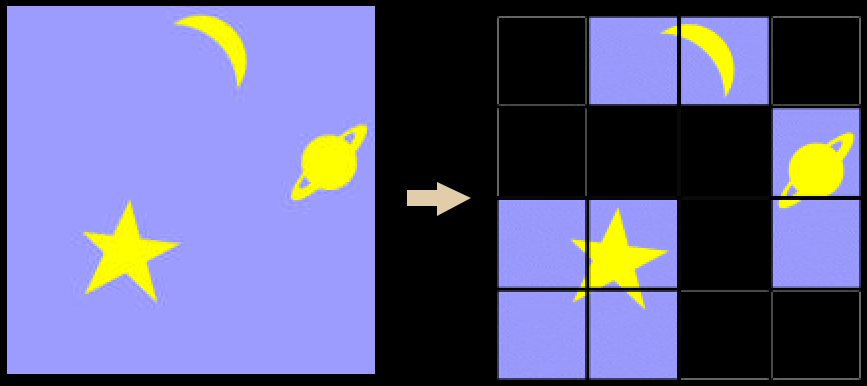
Two Main Parts of This Talk:

- Adaptive Texture Maps in 2 Dimensions
  - Adaptivity and Requirements
  - Data Representation, Sampling, and Generation
- Applications in 3 and 4 Dimensions
  - Volume Rendering
  - Light Field Rendering

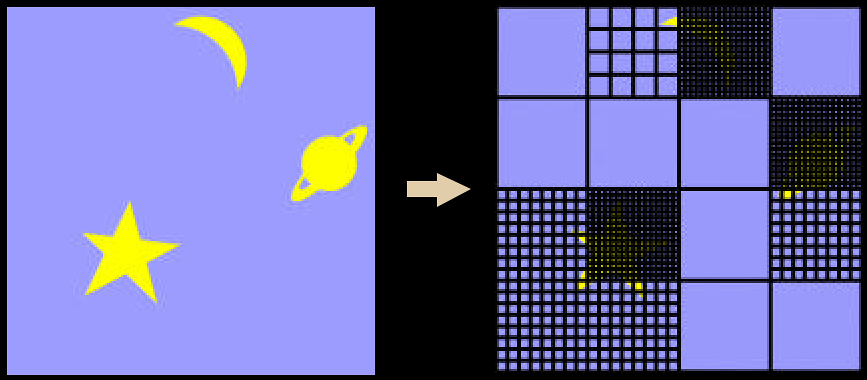
Adaptive Texture Maps in 2 Dimensions

Two Kinds of Adaptivity and Compression:

- Adaptive domain of texture data (lossless compression).



- Locally adaptive resolution of texture data (lossy compression).

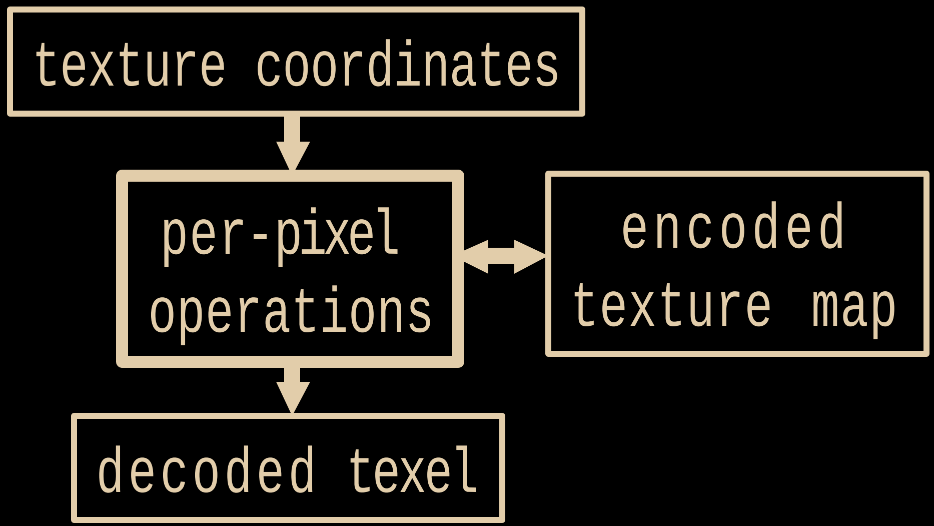


Adaptive Texture Maps in 2 Dimensions

Two Requirements:

- Fast random access to texture data.

  Programmable texturing allows us to decode texture data on-the-fly.



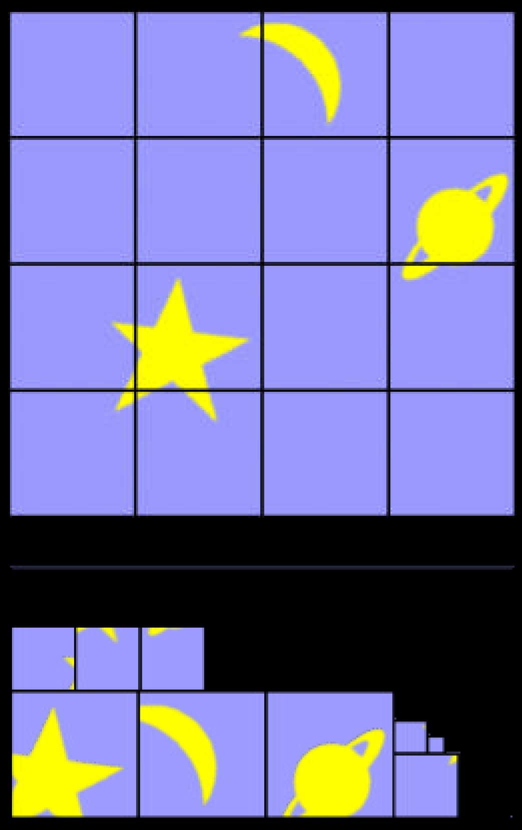
- Two-level data representation.

  Because the ATI Radeon 8500 is limited to one level of dependent texture reads.

Adaptive Texture Maps in 2 Dimensions

Two Levels of the Data Representation:

- Index data (upper level):
  - Each cell/texel of a coarse grid corresponds to one data block.
  - Each cell/texel specifies coordinates and scaling factors of the corresponding data block.
- Packed data (lower level):
  - All data blocks packed into one uniform grid/texture.

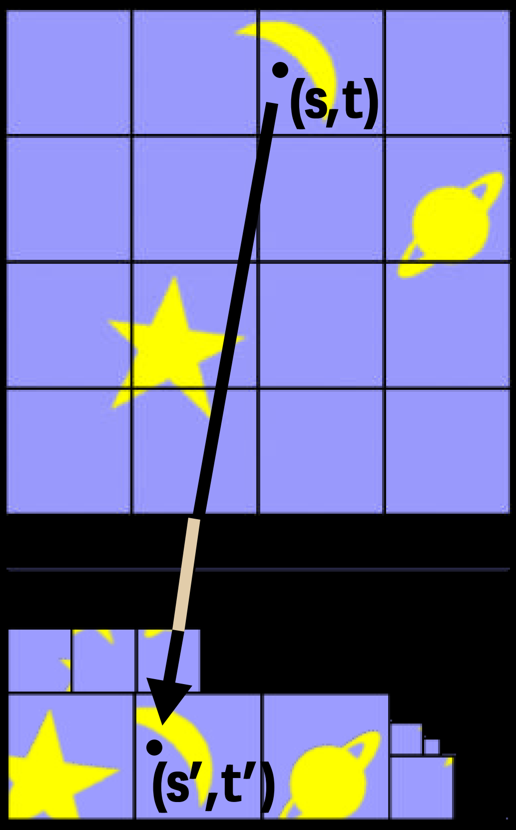


Adaptive Texture Maps in 2 Dimensions

Two Steps of Sampling Adaptive Textures:

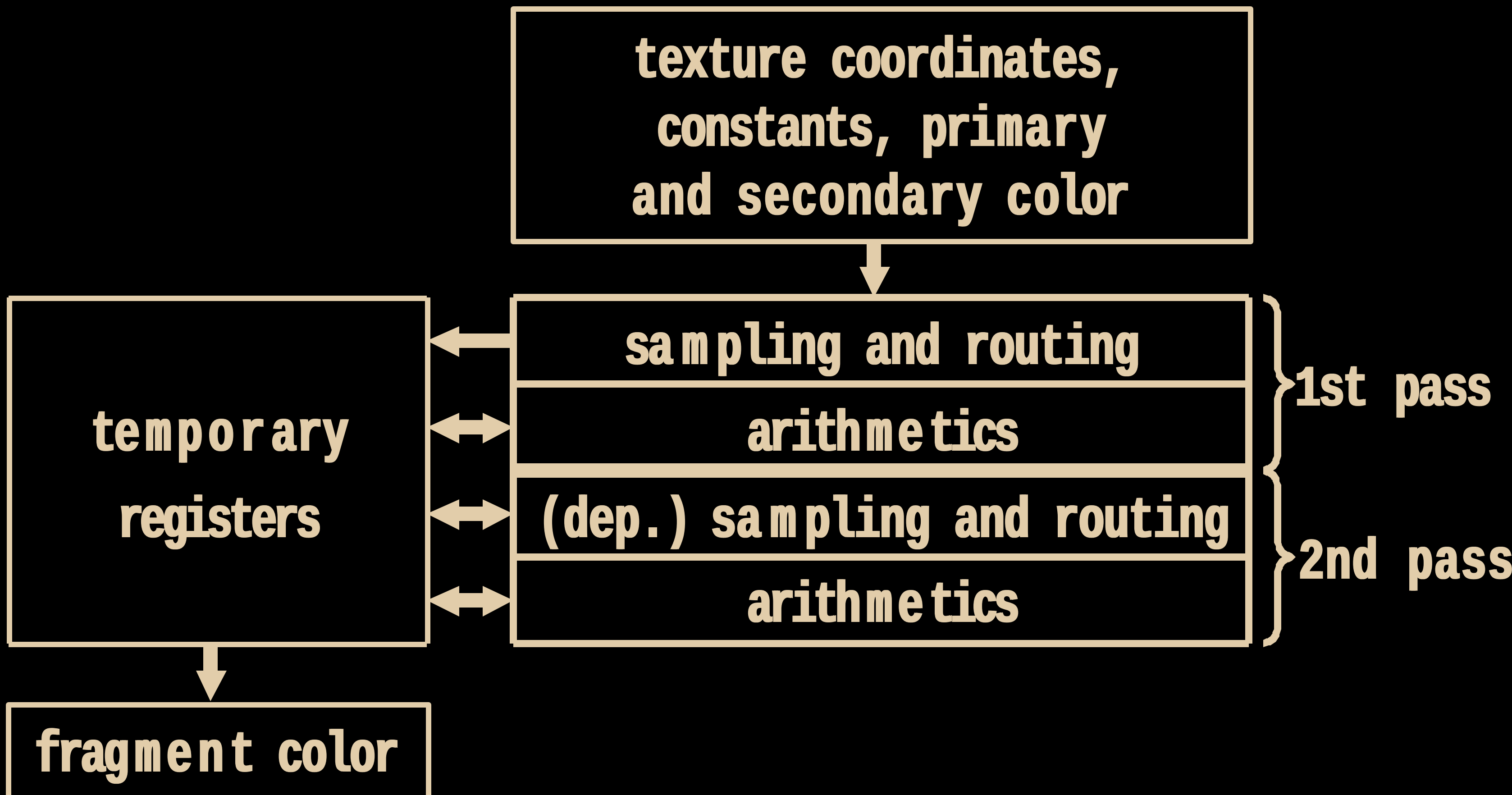
❖ Read index data and calculate coordinates for the second step.

❖ Read and interpolate actual texture data from packed data.



Adaptive Texture Maps in 2 Dimensions

Two Passes of a Fragment Shader Program:



Adaptive Texture Maps in 2 Dimensions

Two Steps of Generating Adaptive Textures:

- Separate downsampling of each data block:
  - Downsampling is repeated for each data block until some error threshold is reached.
  - Scale factors are stored in index data.
  - Special treatment of block boundaries (cont. interpol.!)
- Packing of downsampled data blocks:
  - Simple, non-optimal packing algorithm is sufficient.
  - Coordinates of packed blocks are stored in index data.

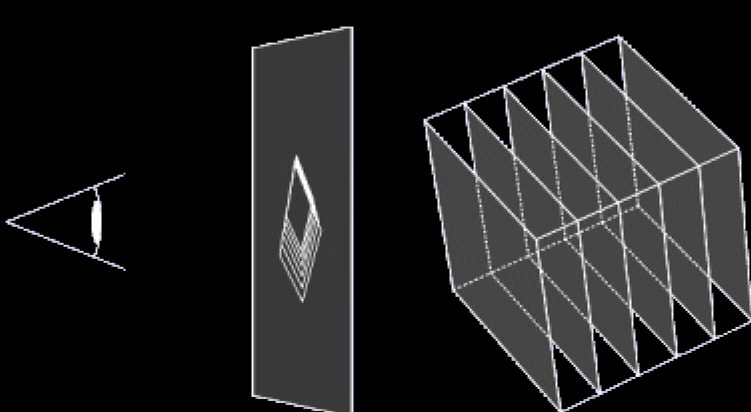
Second Part of the Talk

TWO APPLICATIONS (VARIANTS):

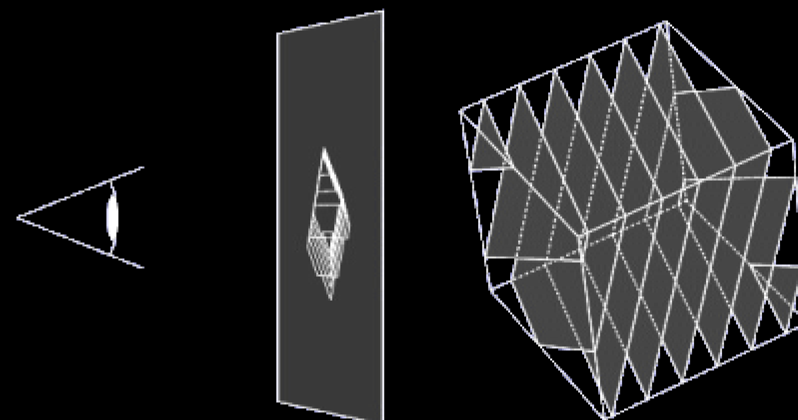
- Volume Rendering (3D Texture Maps)
  (Data from the Stanford volume data archive.)
- Light Field Rendering (4D Texture Maps)
  (Data from the Standford light fields archive.)

Application: Volume Rendering

- Two Kinds of Texture-Based Volume Rendering:



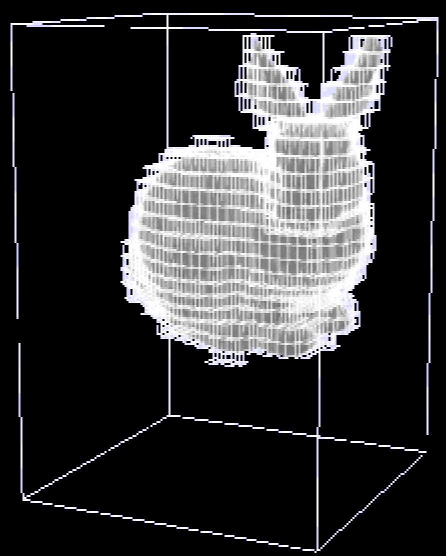
object-aligned
(2d textures)



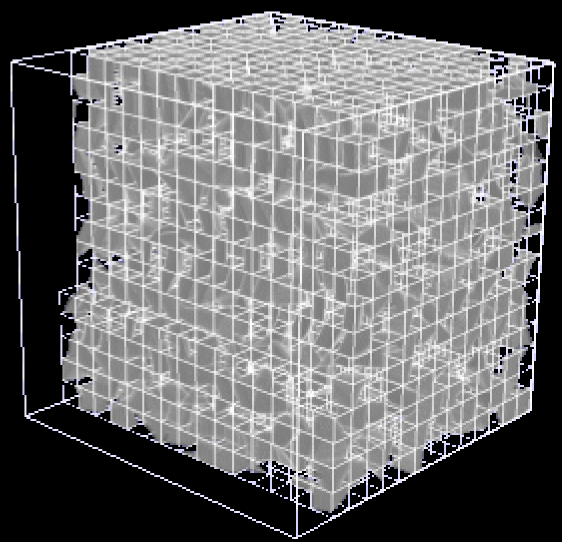
viewplane-aligned
(3d textures)

Application: Volume Rendering

- Two Internal 3D Texture Maps For Adaptive 3D Texture Maps:



index data: 32^3 cells



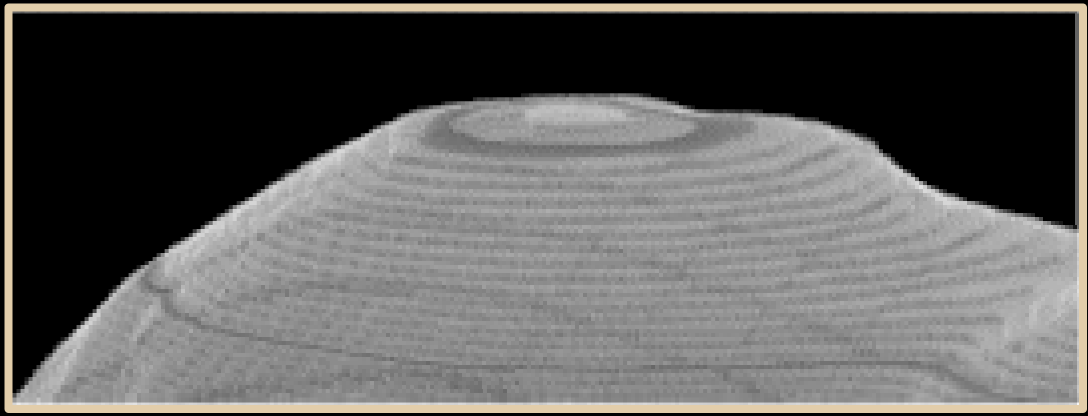
packed data: 256^3 voxels

(32^3 cells $\times 16^3$ voxels/cell = 512^3 voxels)

Application: Volume Rendering

Two Kinds of Artifacts:

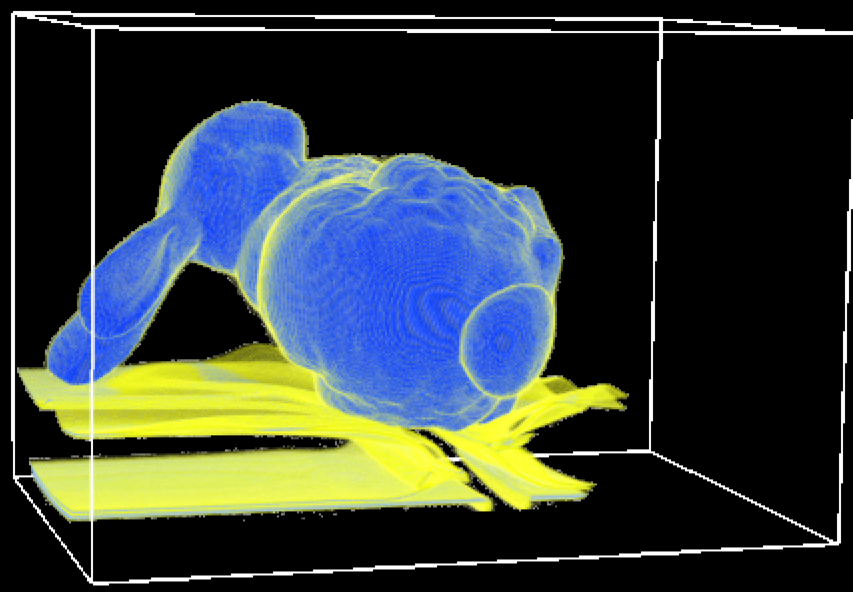
- Sampling error in data set.
- Discontinuous boundaries between data blocks because of fixed-point arithmetics in fragment shader programs.



Application: Volume Rendering

Two Steps to Vector Quantization:

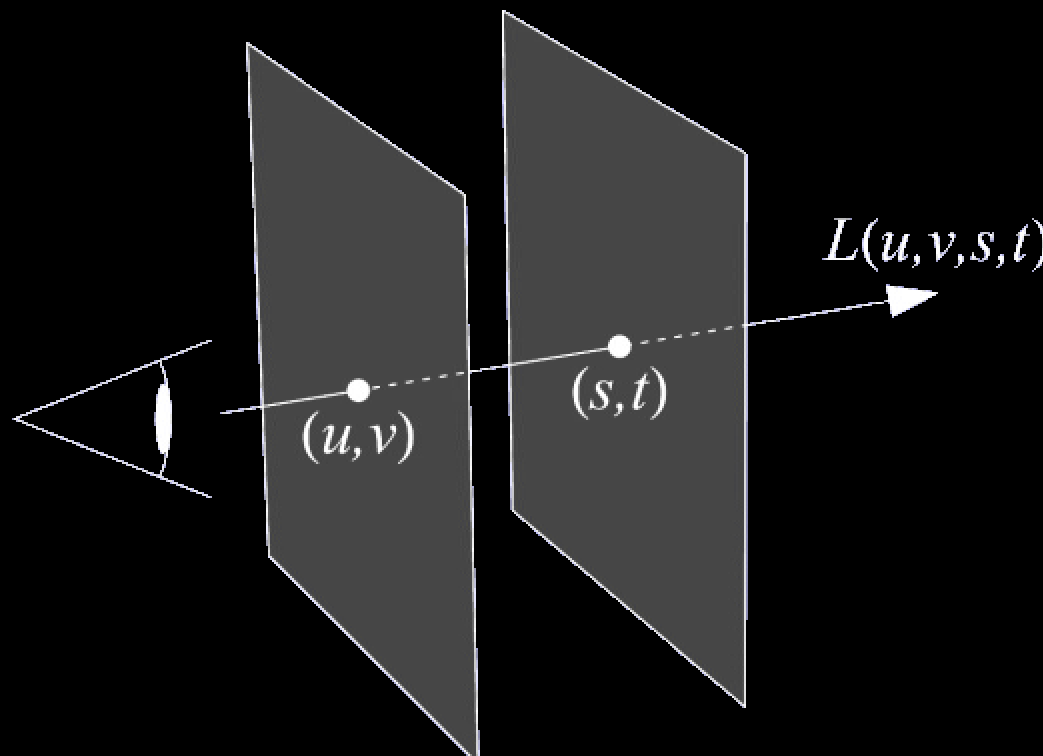
- Use few (256), tiny data blocks (2^3 voxels).
- Use nearest-neighbor interpolation in packed data.



(See "Texture Compression" in EUROGRAPHICS 2002, Tutorial T4.)

Application: Light Field Rendering

- Two Pairs of Coordinates:

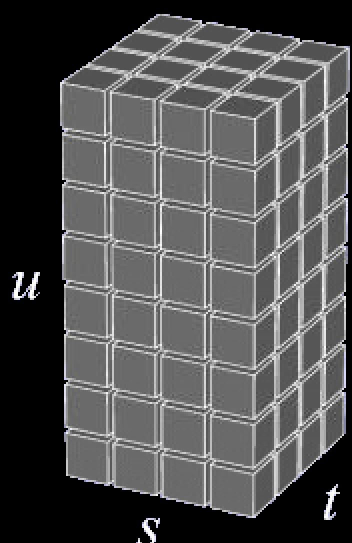


(Levoy, Hanrahan: Light Field Rendering, SIGGRAPH '96.)

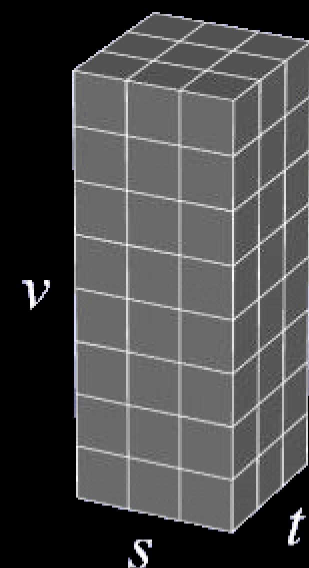
Application: Light Field Rendering

Two-Level Data Represenation:

- Each data block covers several values of s and t, one value of u and all values of v.



index data



one data block

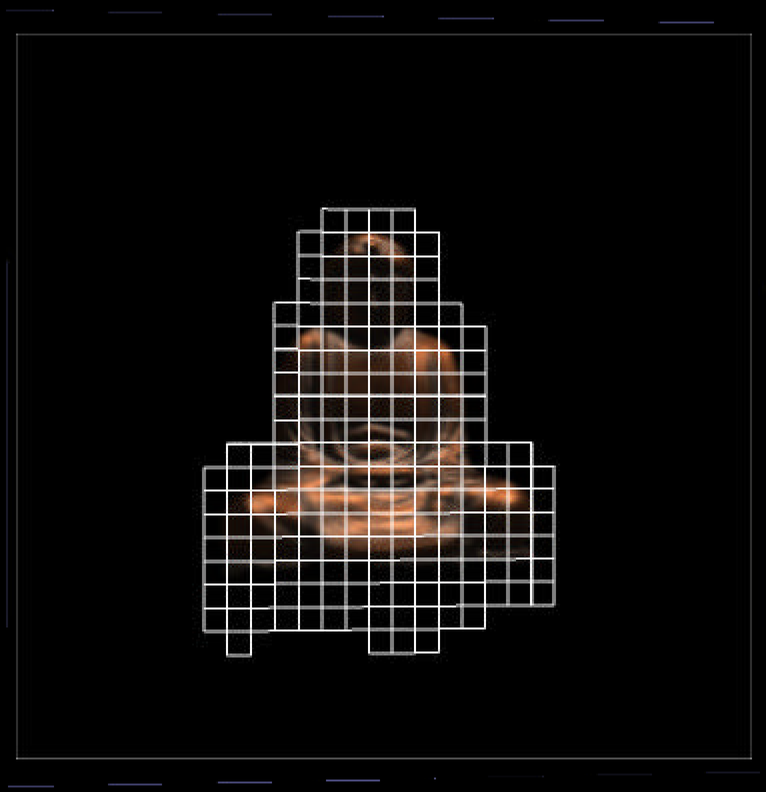
Application: Light Field Rendering

Two Trilinear Interpolations For One Quadrilinear Interpolation:

- Trilinear interpolation of L(floor(u), v, s, t).
- Trilinear interpolation of L(ceiling(u), v, s, t).
- Linear interpolation of results with weights ceiling(u) - u and u - floor(u) gives L(u, v, s, t).

Application: Light Field Rendering

Two Examples:



Conclusions

Two Questions:

- How useful are adaptive texture maps?
  - Depends very much on the texture data.
  - Very useful for data with strongly varying resolution.
  - Also useful for data with large empty regions.
  - Vector quantization is useful for volume rendering.
- Should they be implemented in hardware?
  - No, because programmable graphics hardware will soon be good enough.

Future Work

Two Areas:

- Exploiting new graphics hardware:
  - Floating-point precision,
  - Combination with other per-pixel computations,
  - Deeper hierarchies with more dependent texture reads.
- Exploring more fields of application:
  - Normal maps, environment maps, shadow maps, …
  - BRDFs, multi-dimensional transfer functions, …

Two More Things to Say:

- Thank you!
- Any questions?