

Image-based Modeling and Rendering: 9. The Lumigraph & Light Field Rendering

I-Chen Lin, Assistant Professor
Dept. of CS, National Chiao Tung University

Outline

- How to represent an object with images?
- Construct the plenoptic function:
 - The lumigraph
 - Light field rendering

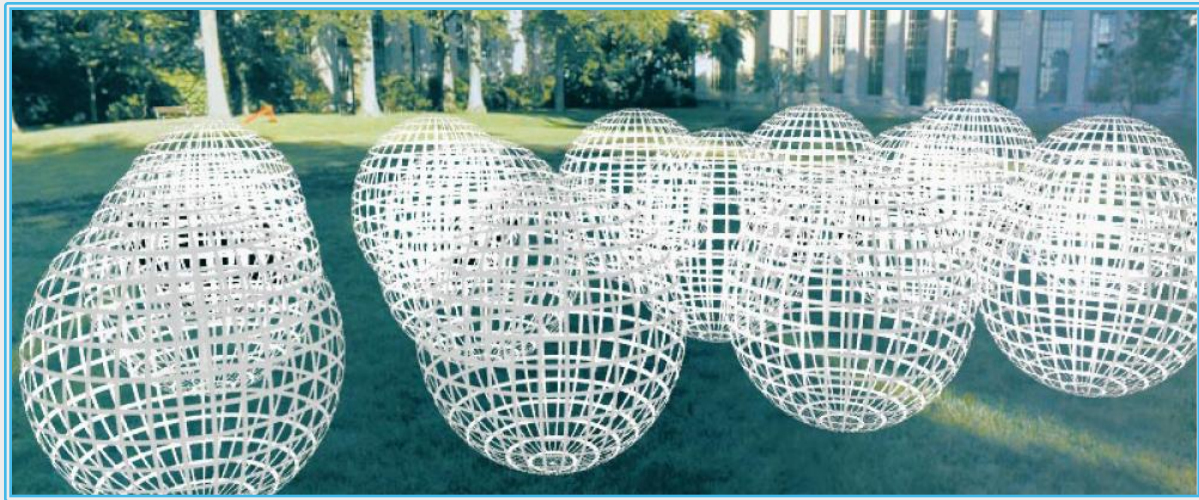
Ref:

- Image-based Modeling and Rendering, SIGGRAPH'99 course notes.
- M. Levoy, P. Hanrahan, "Light Field Rendering", Proc. SIGGRAPH'96, pp.31 - 42, 1996.
- S. J. Gortler, R. Grzeszczuk, R. Szeliski, M.F. Cohen, "The Lumigraph", Proc. SIGGRAPH 96, pp.43 - 54, 1996.

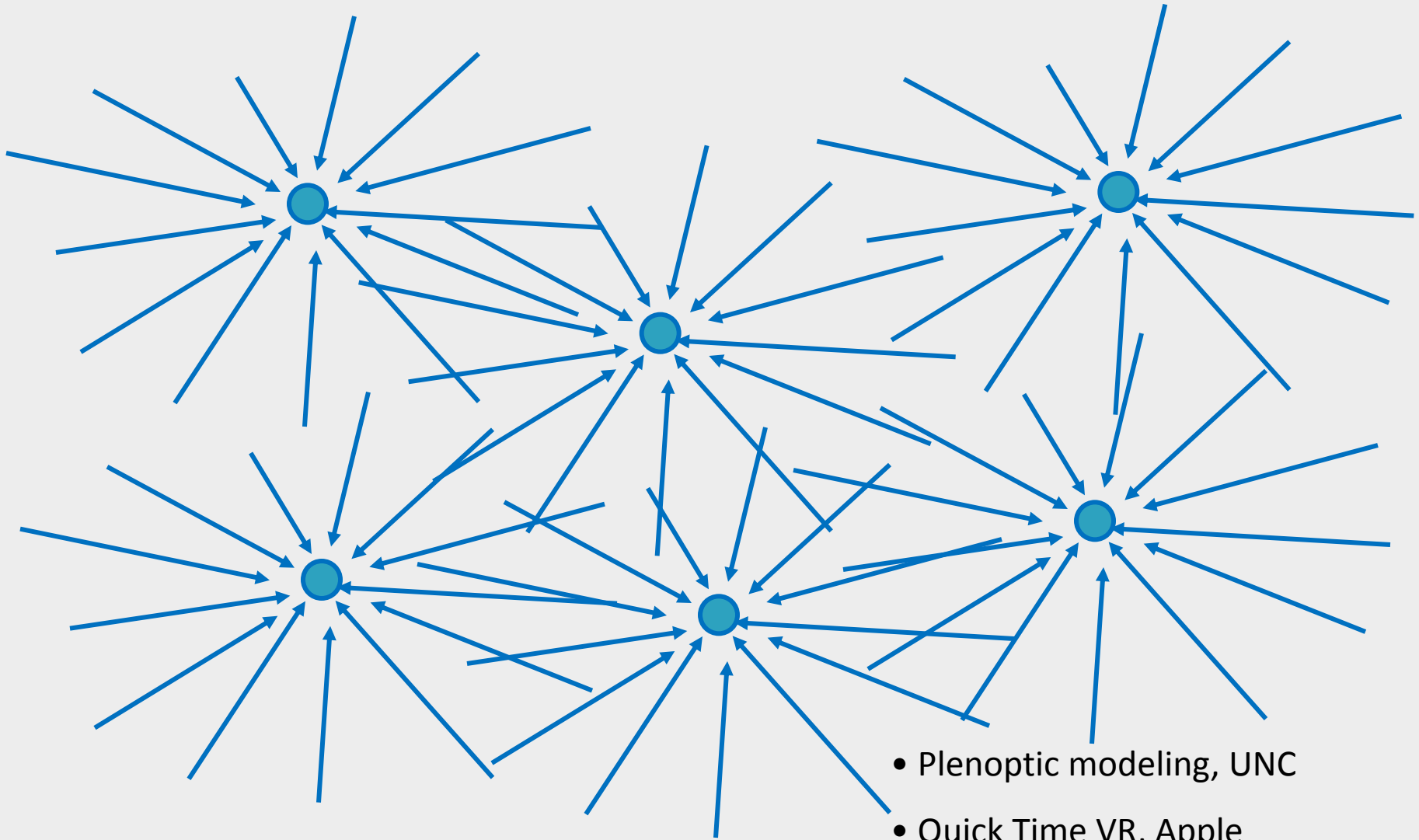
The Plenoptic Function

- Reconstruct every possible view, at every moment, from every position, at every wavelength

$$P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$$

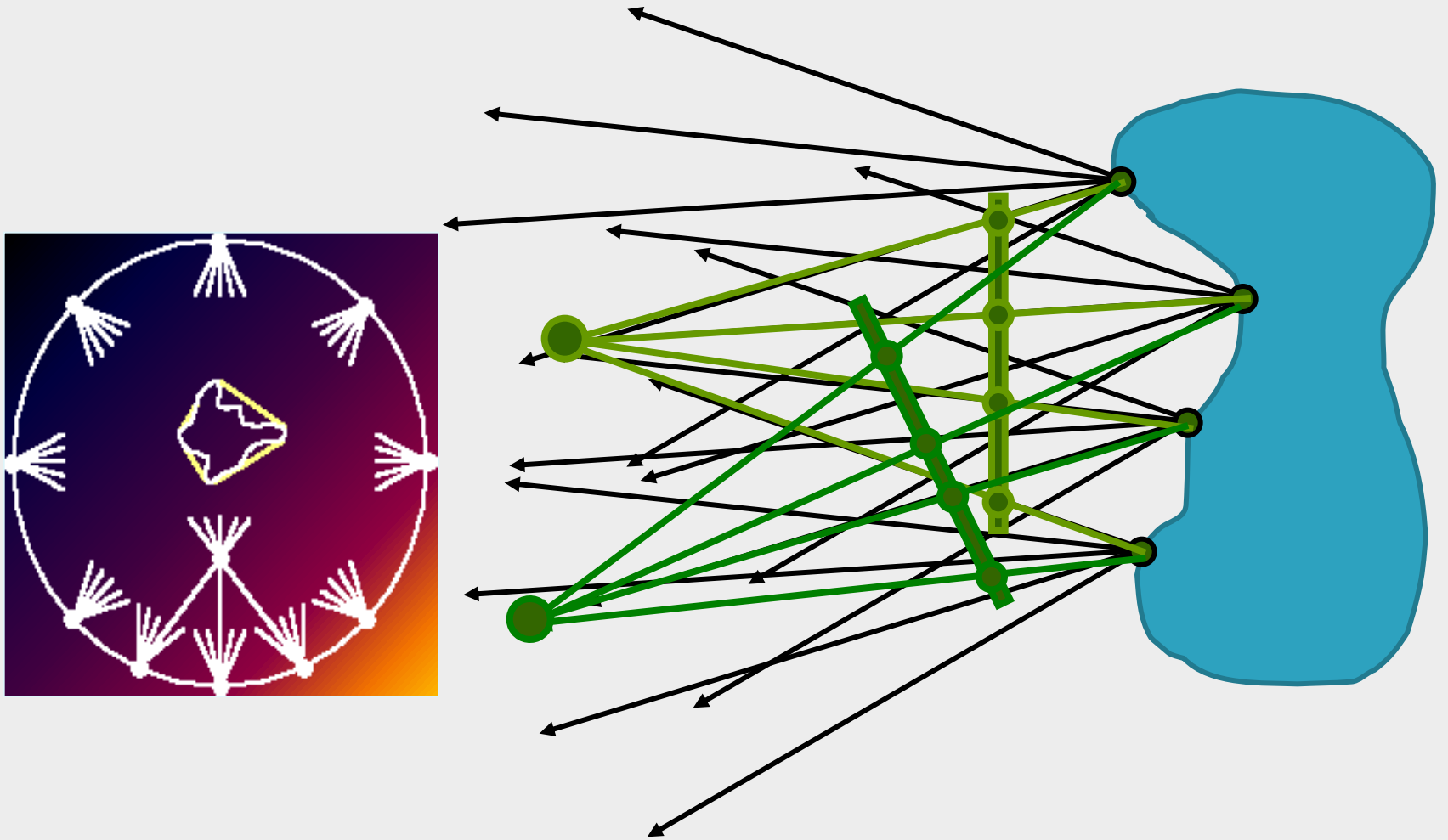


Sampling The Plenoptic Function



Sampling The Plenoptic Function

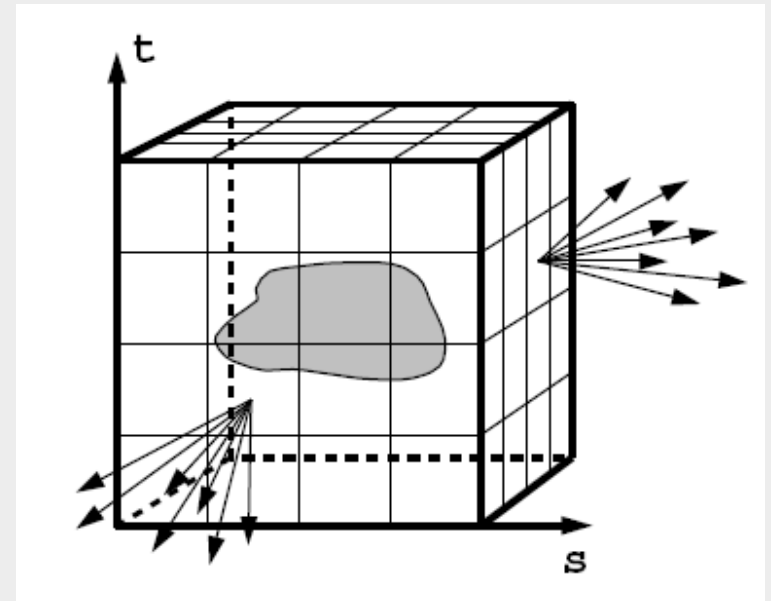
- How to reconstruct the plenoptic function for an object?



Only need plenoptic surface

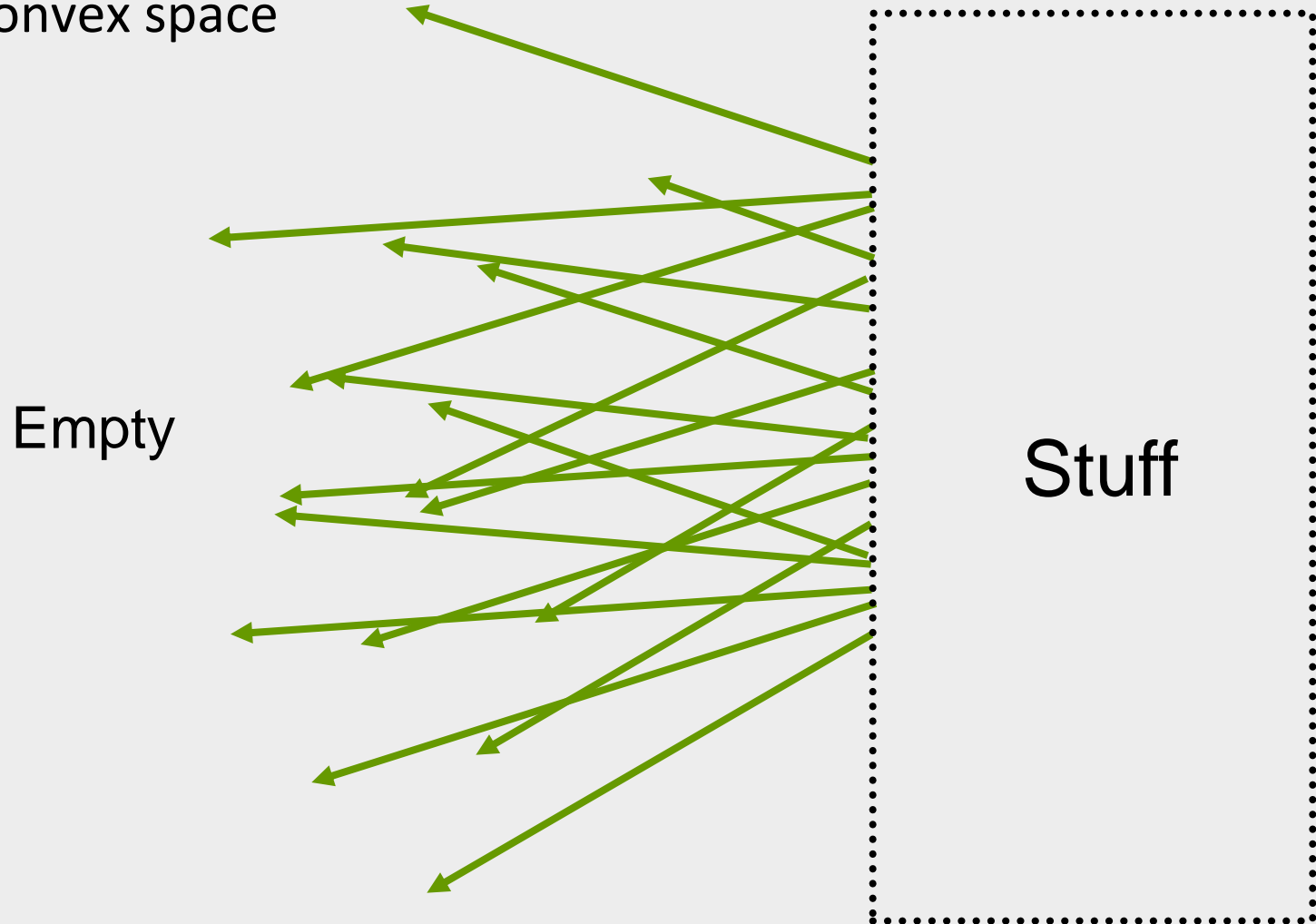
- In “free space” (no occluders) the dimensionality is reduced from 5D to 4D.
 - Exterior of the convex hull of an object
 - Interior of an environment
- An image is a 2D slice of the 4D light field.

The surface of a cube holds all the radiance information due to the enclosed object.



Lumigraph / Lightfield

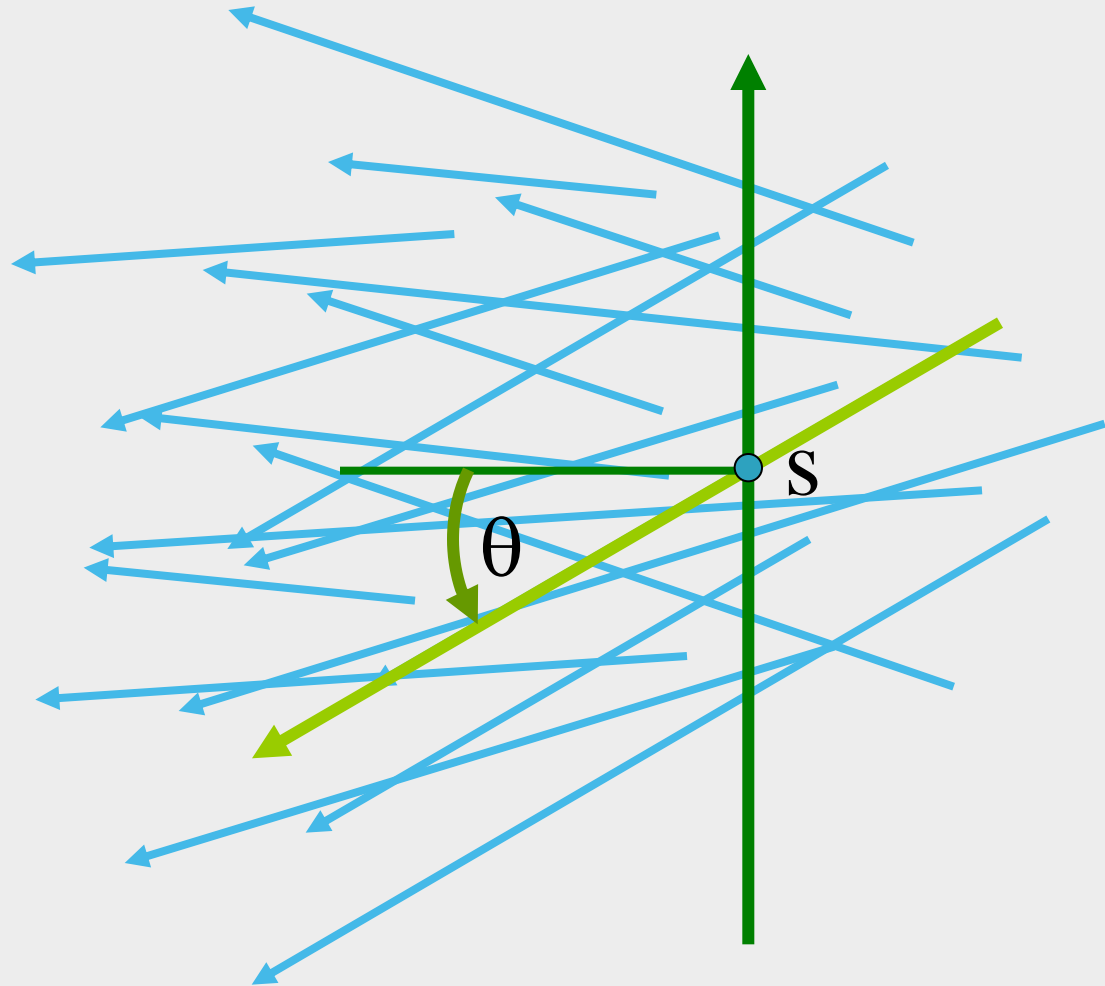
- Outside convex space



4D

Lumigraph / Lightfield

- 2D position
- 2D direction

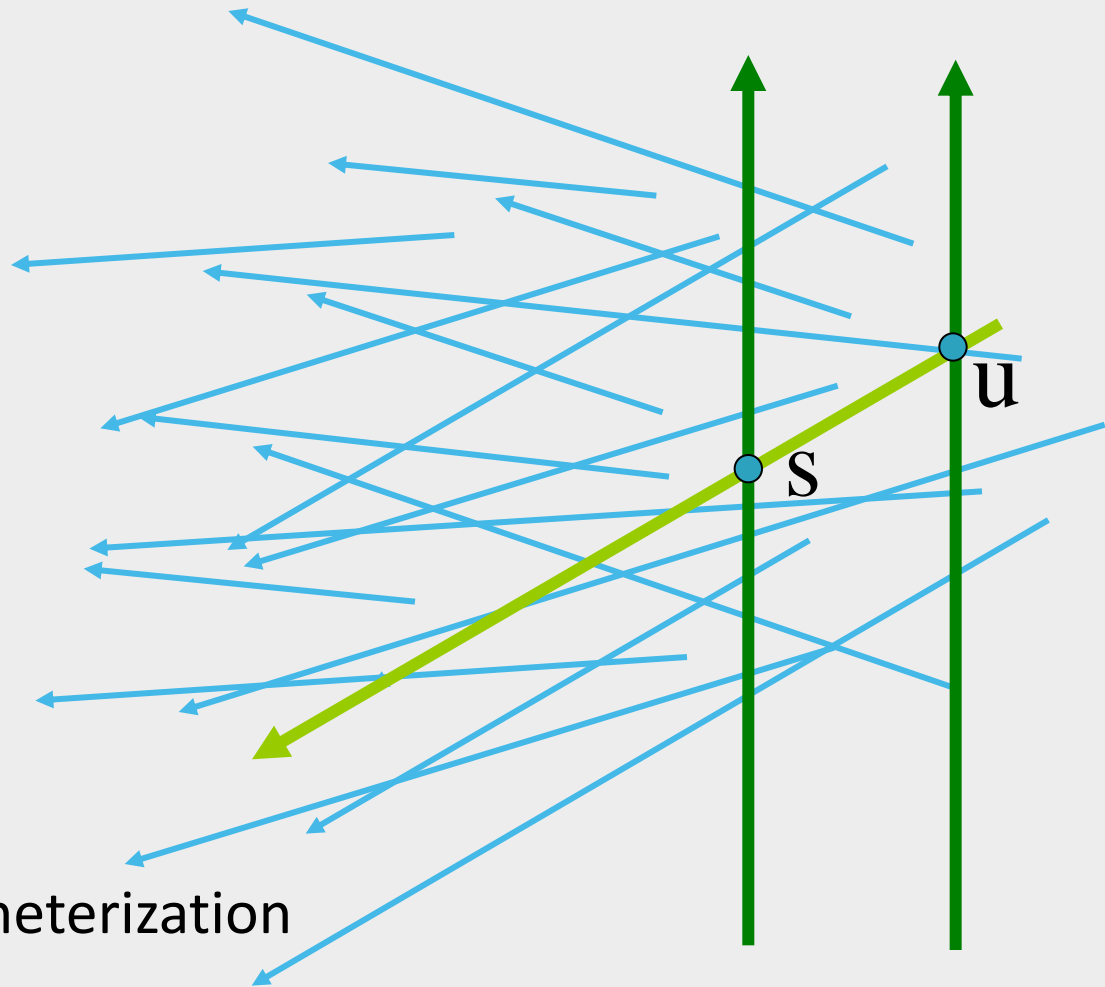


Lumigraph / Lightfield

■ 2D position

■ 2D position

■ 2 plane parameterization

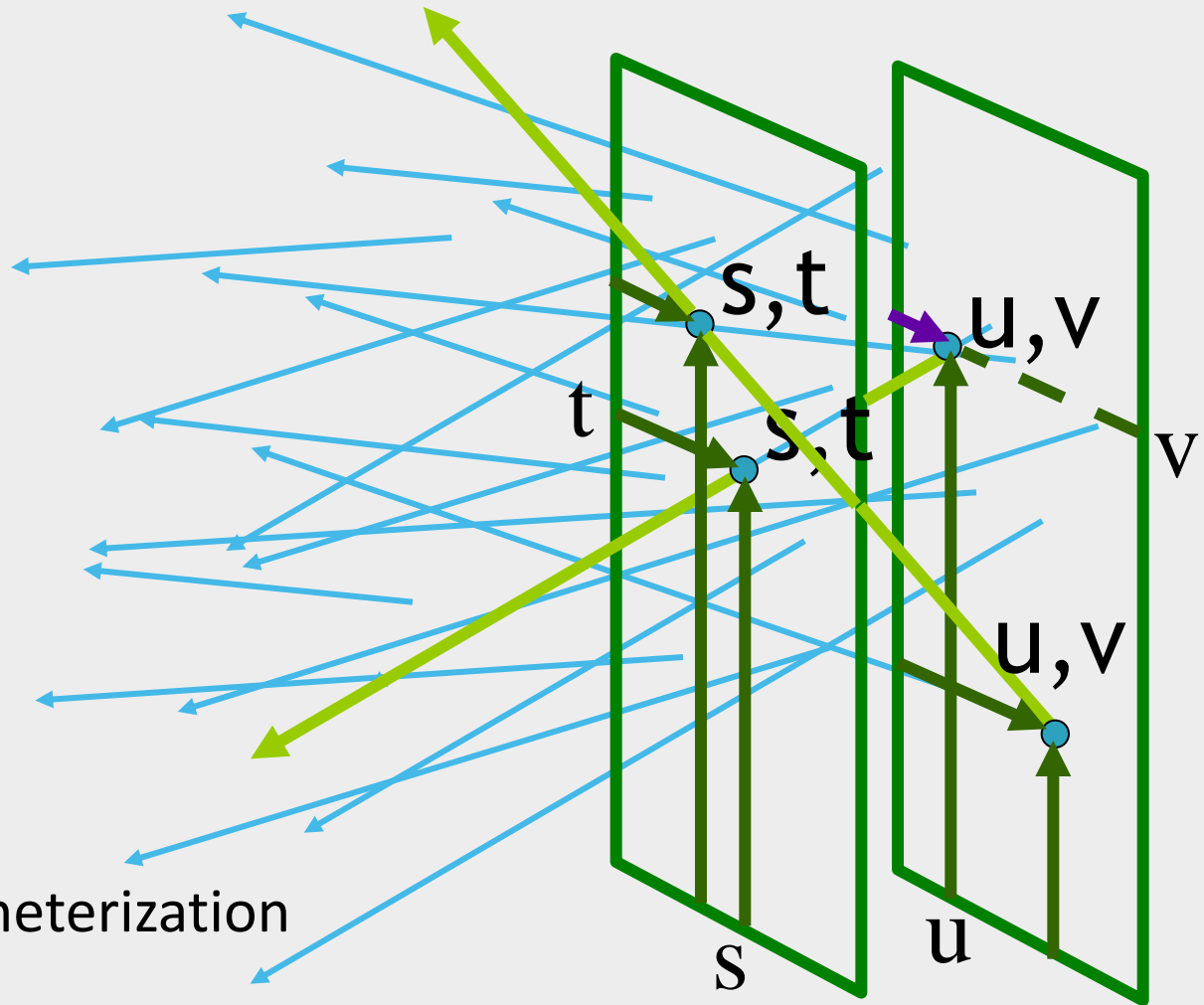


Lumigraph / Lightfield

■ 2D position

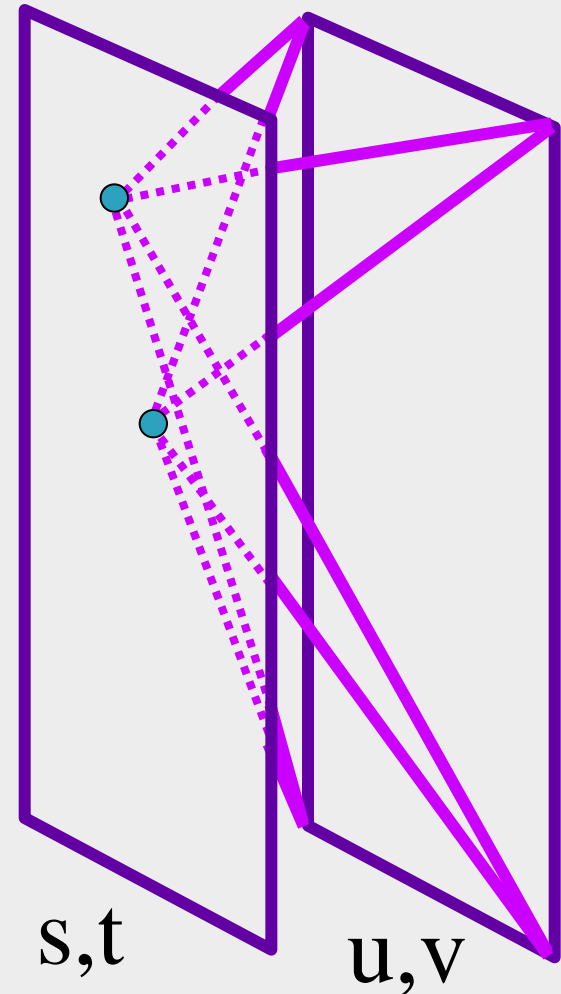
■ 2D position

■ 2 plane parameterization



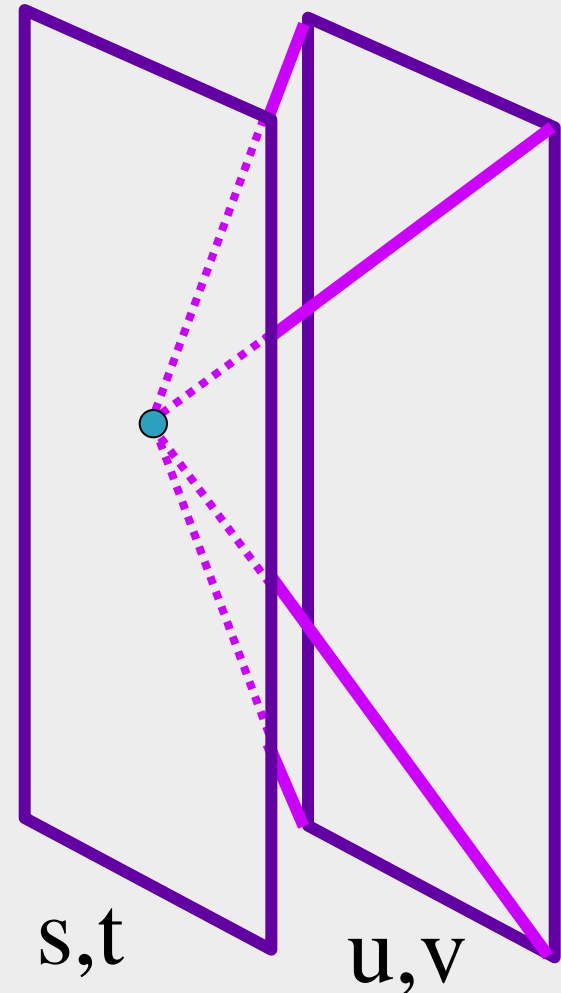
Lumigraph / Lightfield

- Hold s, t constant
- Let u, v vary
- An image

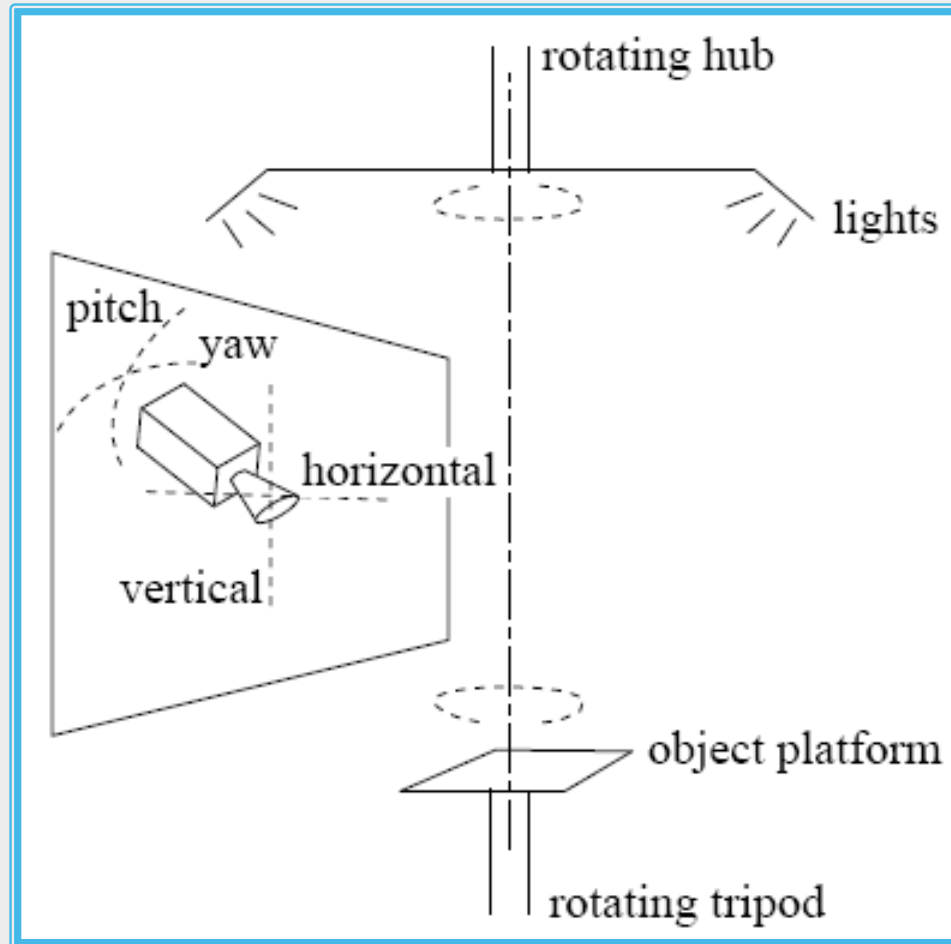


Lightfield - Capture

- Idea 1
 - Move camera carefully over s,t plane
 - Gantry
 - see Lightfield paper



Lightfield - Capture



Stanford multi-camera array

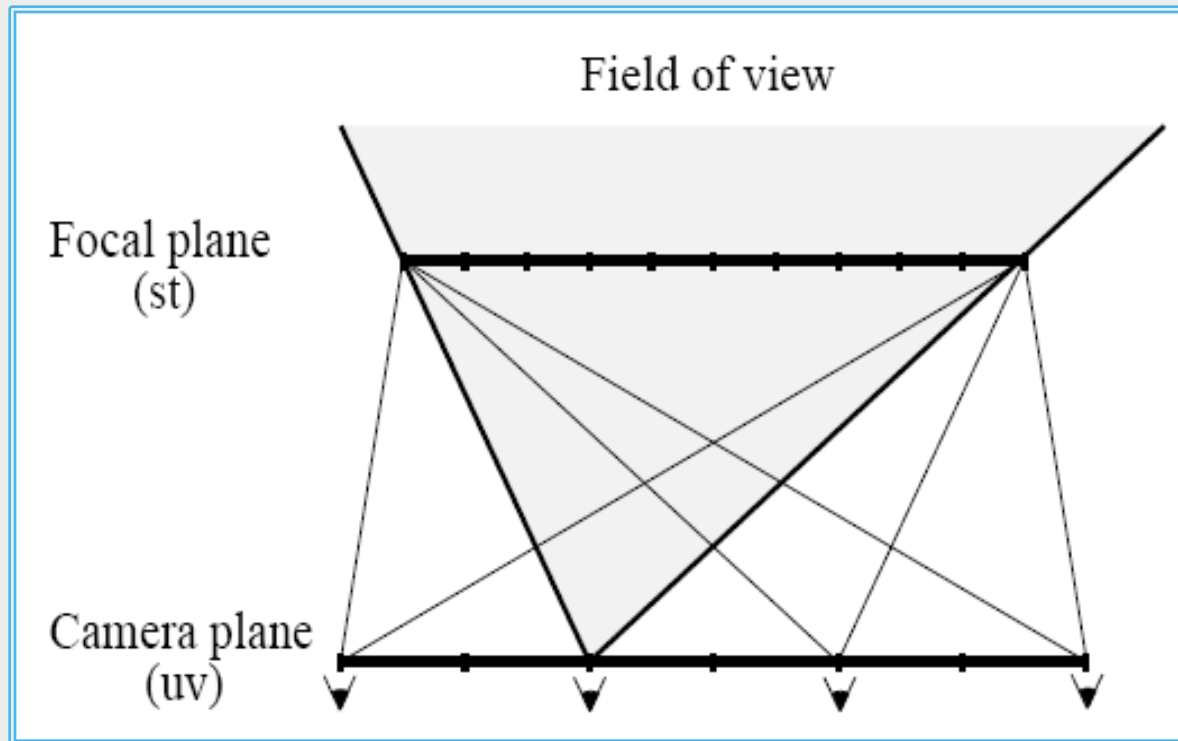


- 640×480 pixels \times
30 fps \times 128 cameras
- synchronized timing
- continuous streaming
- flexible arrangement

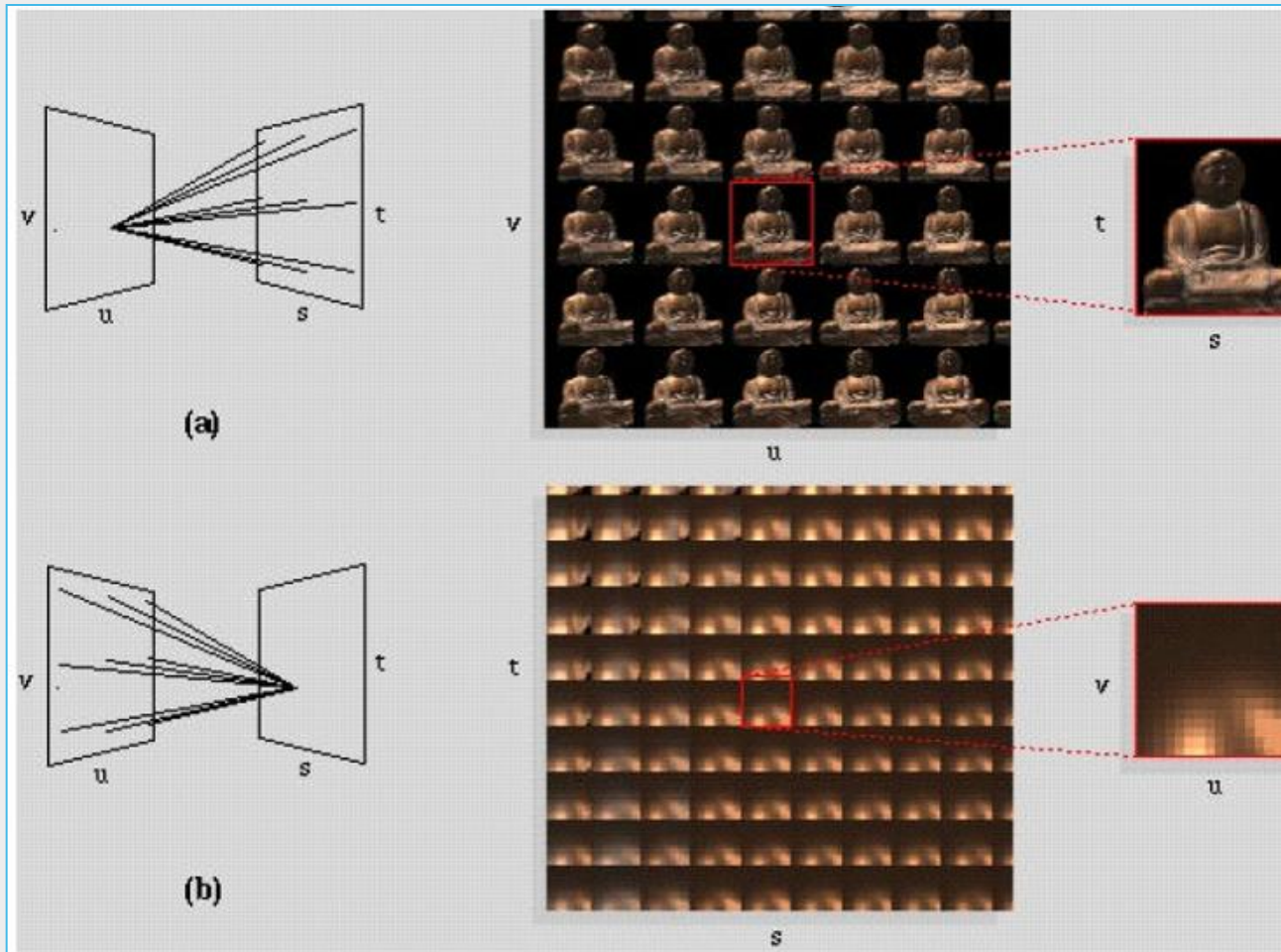


Lightfield - Capture

Utilizing sheared perspective transformation



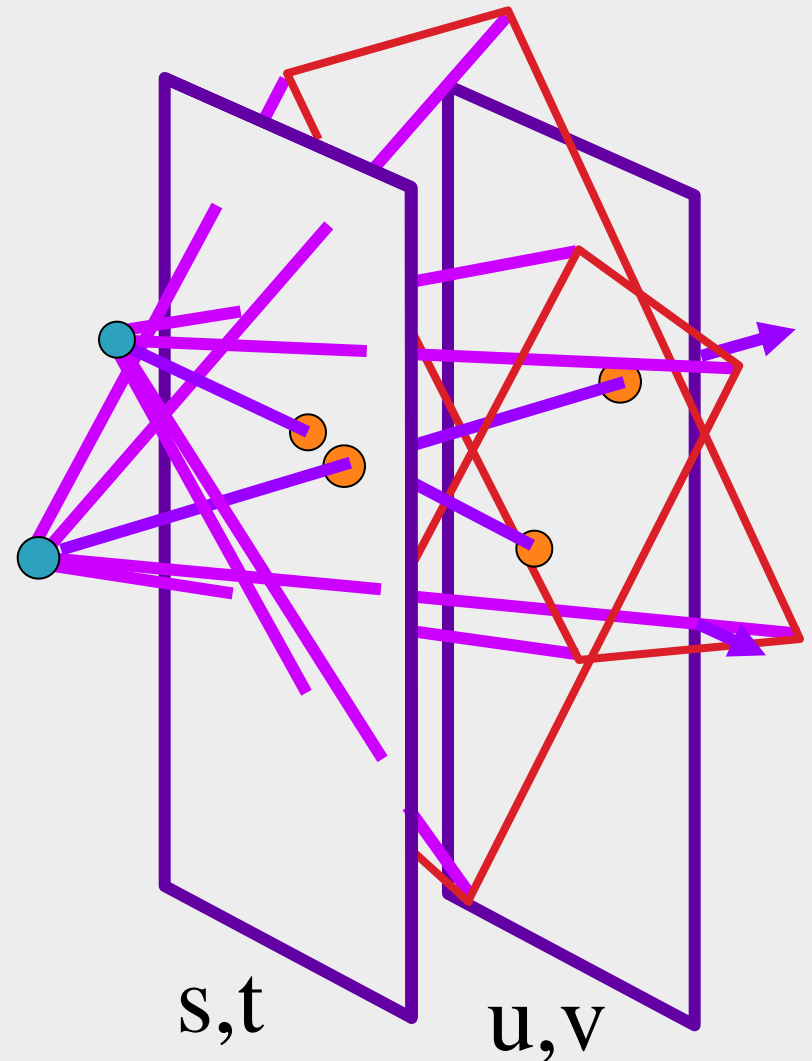
Lumigraph / Lightfield



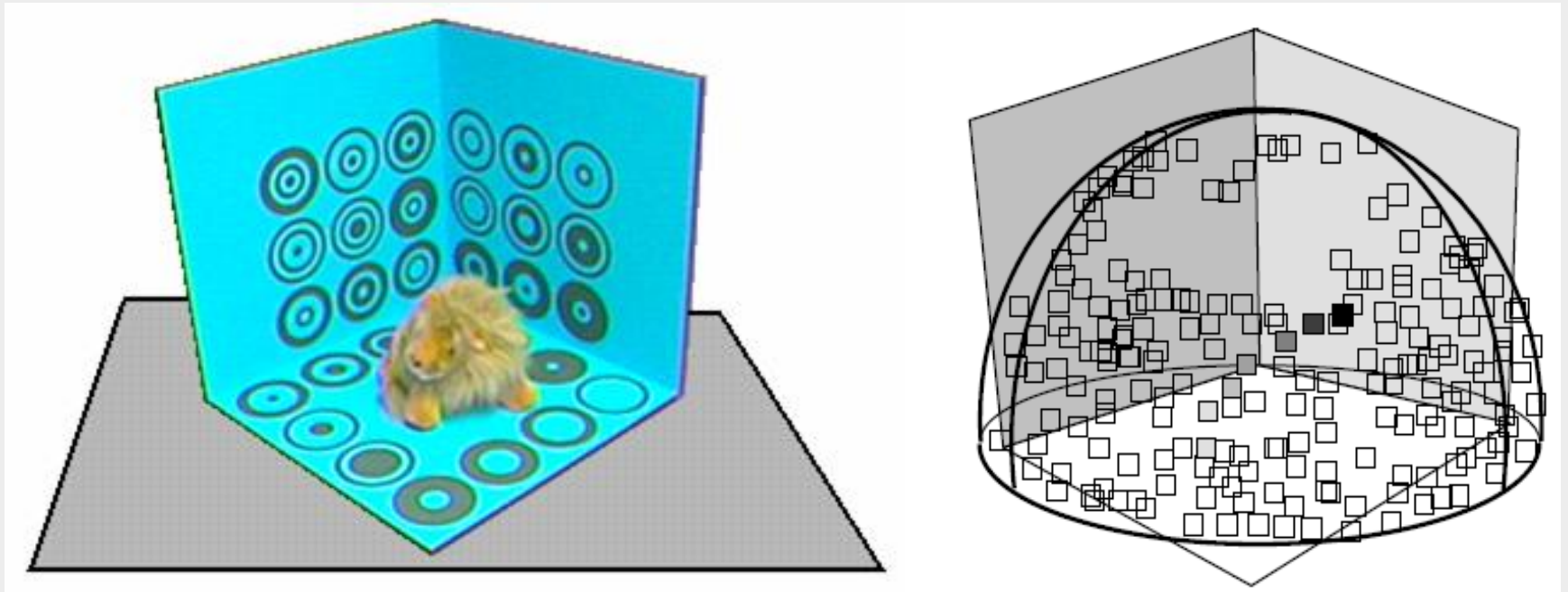
The notation of st and uv planes are exchanged in papers of Lumigraph and LightField

Lumigraph - Capture

- Idea 2
 - Move camera anywhere
 - Camera intrinsics assumed calibrated
 - Camera pose recovered from markers
- Rebinning
 - see Lumigraph paper

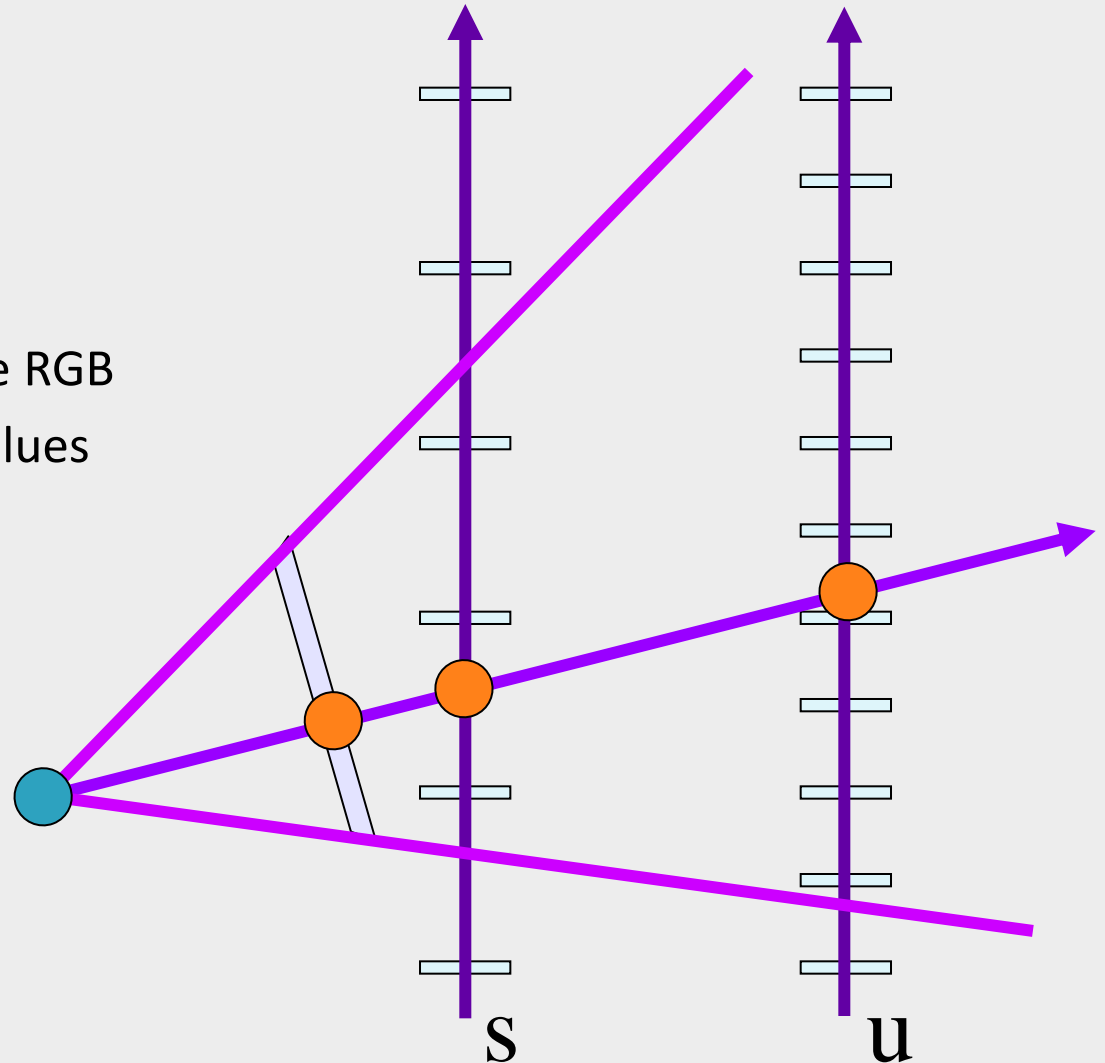


Lumigraph - Capture



Lumigraph/Lightfield - Rendering

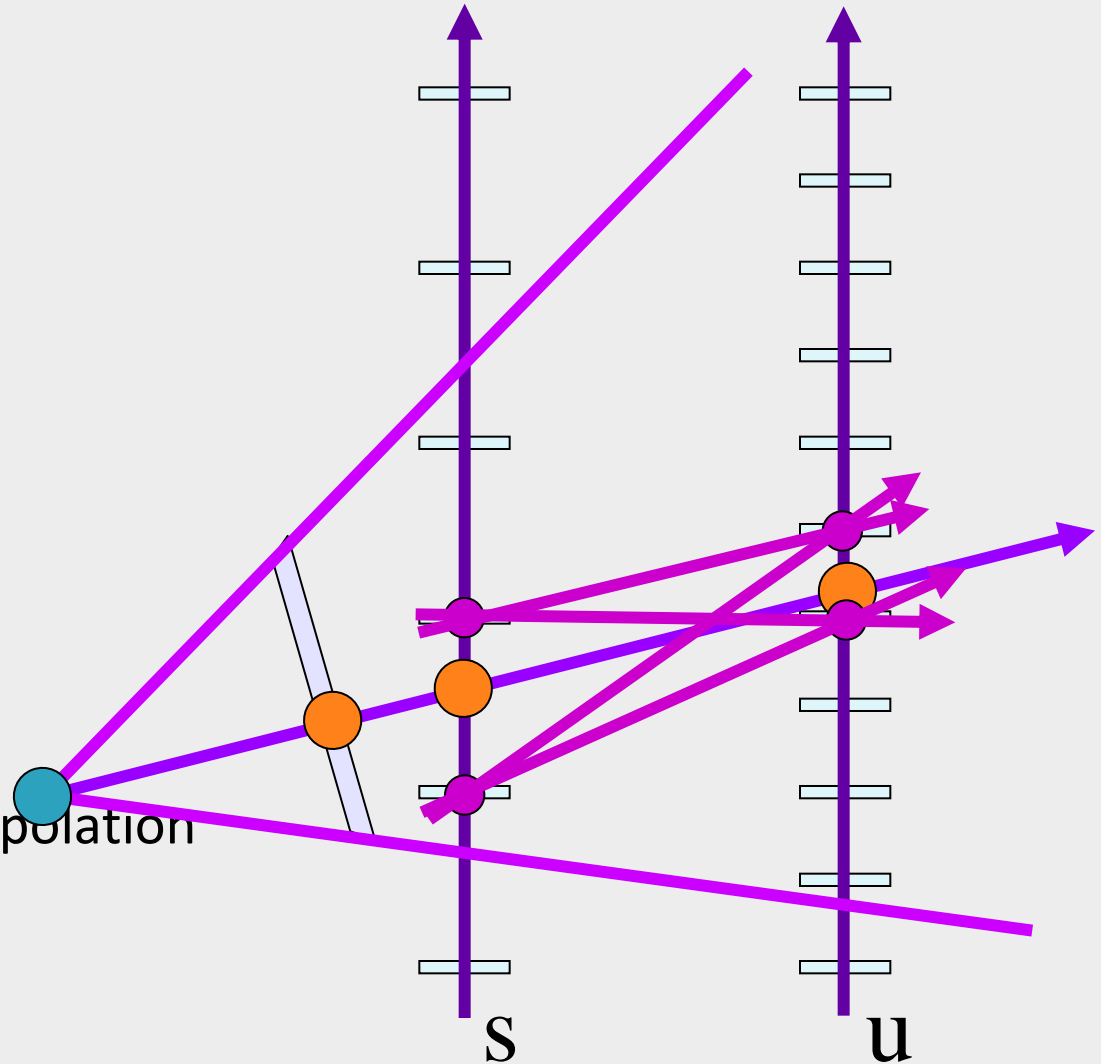
- For each output pixel
 - determine s, t, u, v
 - either
 - use closest discrete RGB
 - interpolate near values



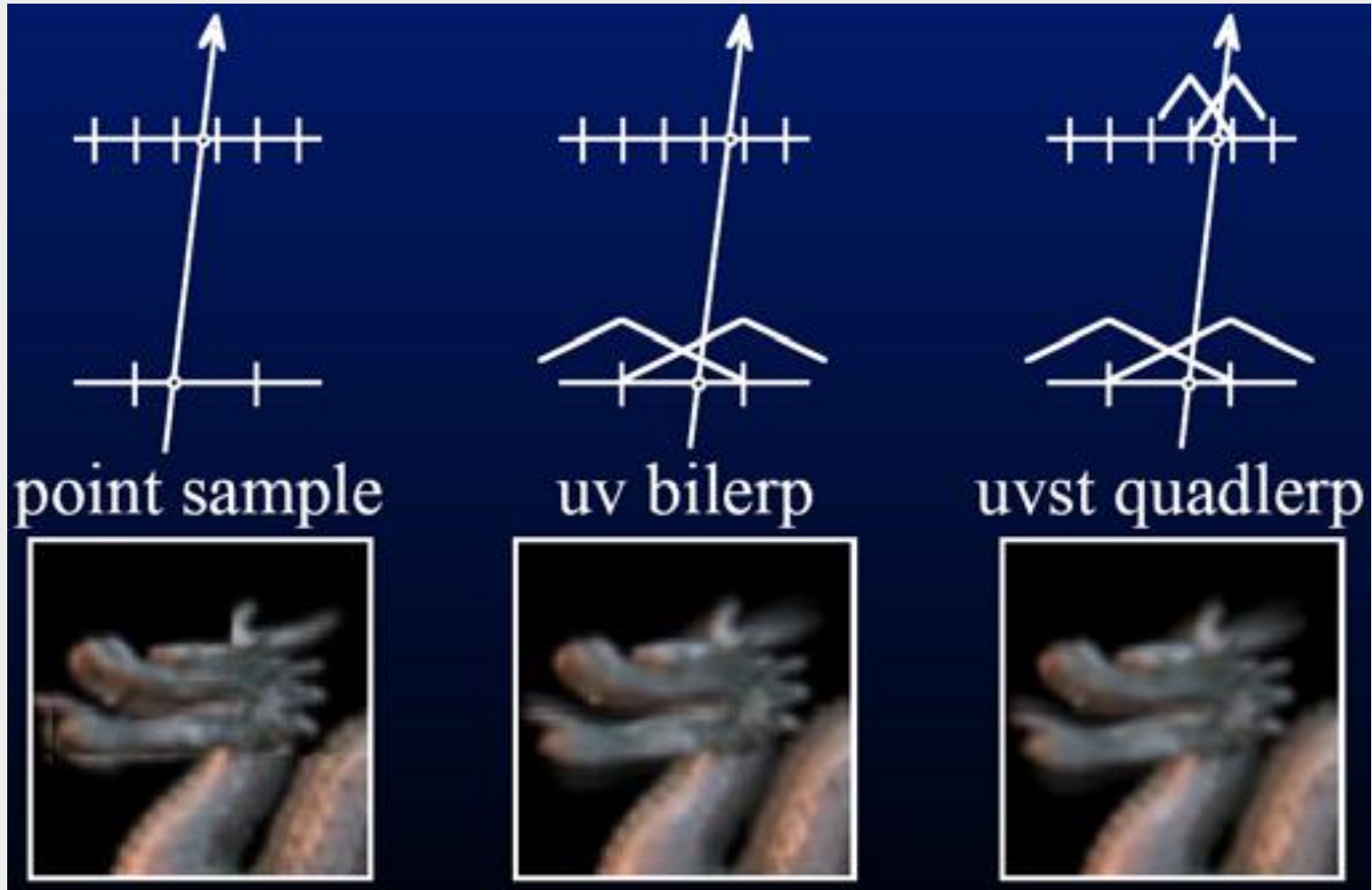
Lumigraph/Lightfield - Rendering

- Nearest
 - closest s
 - closest u
 - draw it

- Blend 16 nearest
 - quadrilinear interpolation



Lumigraph/Lightfield - Rendering

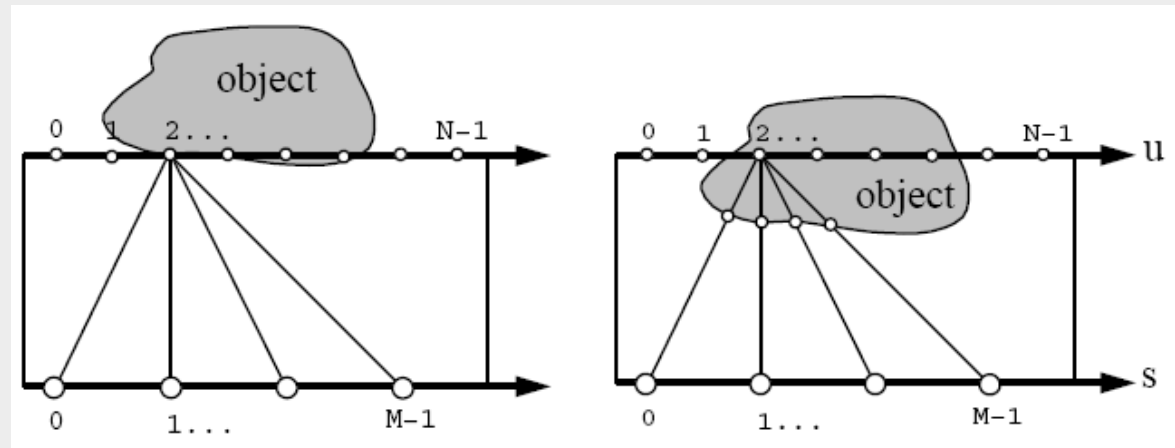


Other issues

- How many samples are sufficient?
- Compression of large image sets.
- Improvement of synthesized image quality.

Resolution

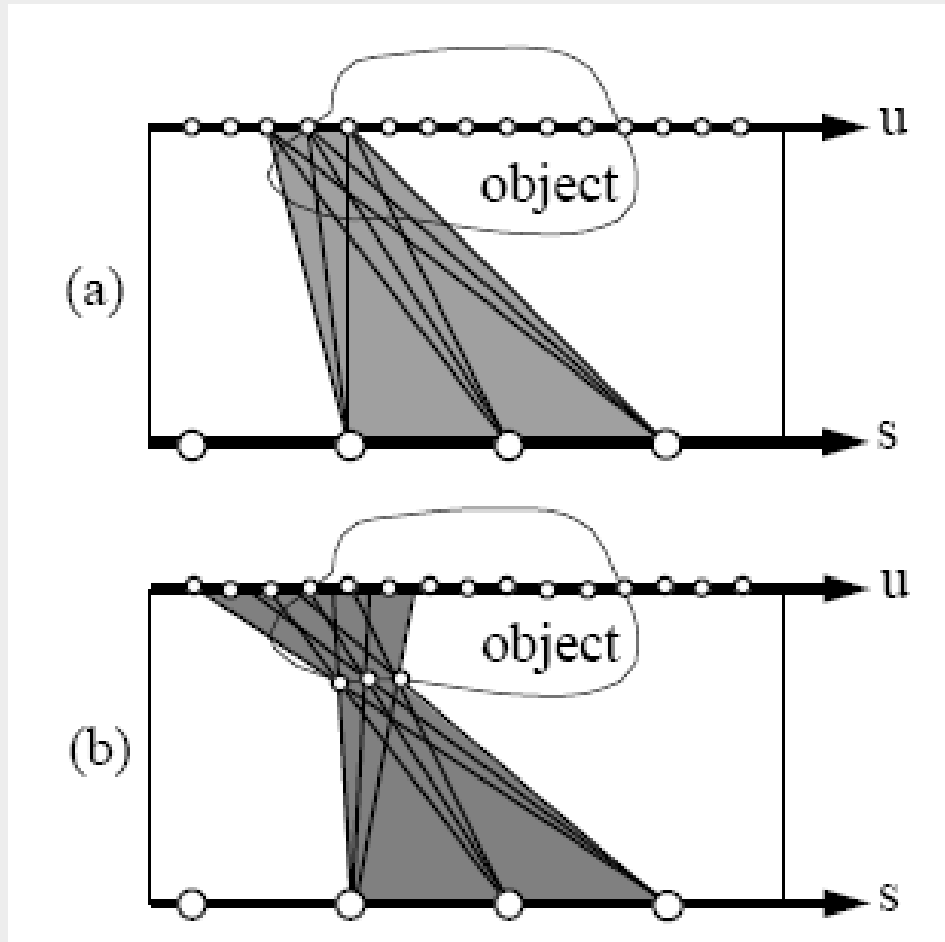
- In Lumigraph paper, they suggested that the far plane needs to have resolution of the desired image.
- The near plane represents the change of view direction; the resolution can be lower.
- Far: 128 to 512
- Near: 16 to 64



Light Field Compression

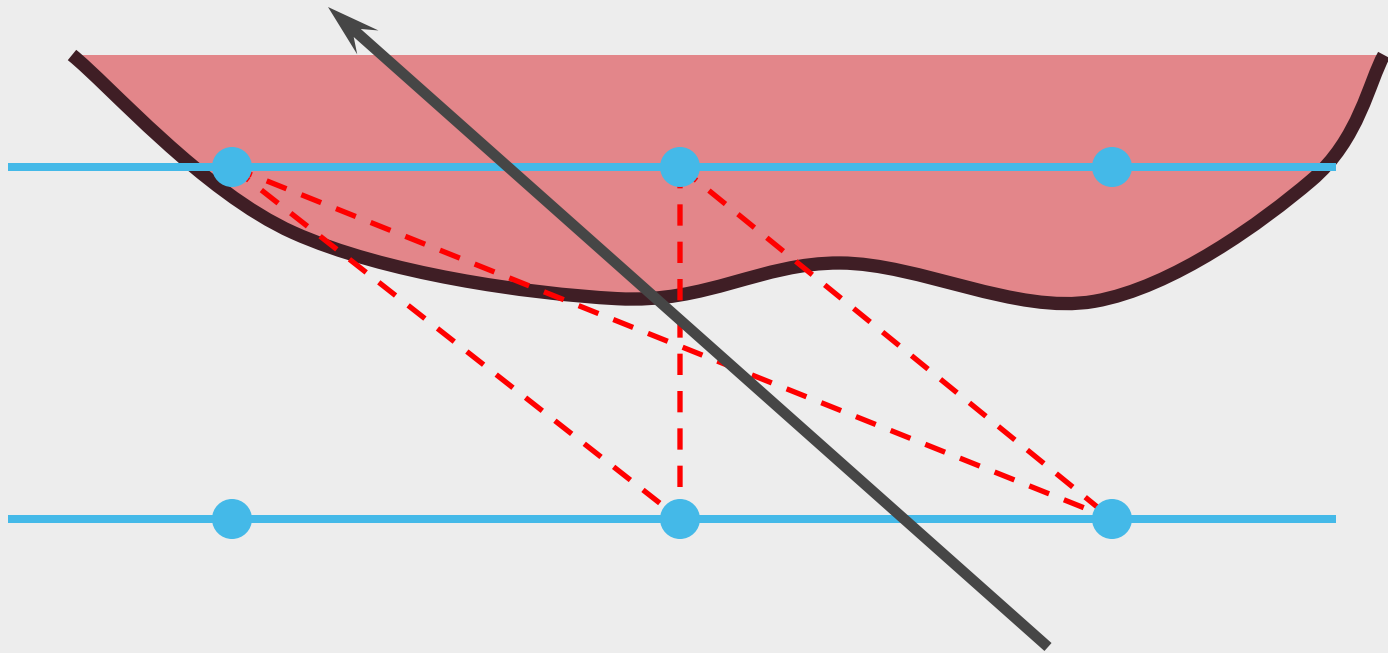
- Compressing based on **vector quantization**
 - Preprocessing: build a representative codebook
 - Each tile in lightfield represented by index
 - 24:1 compression
- Also applying **entropy coding**, e.g. Lempel-Ziv, Huffman coding.
 - 5:1 compression
- 120:1 (for 1.6G MB image sets)
- (Lumigraph used other JPEG/MPEG-like compression)

Depth correction



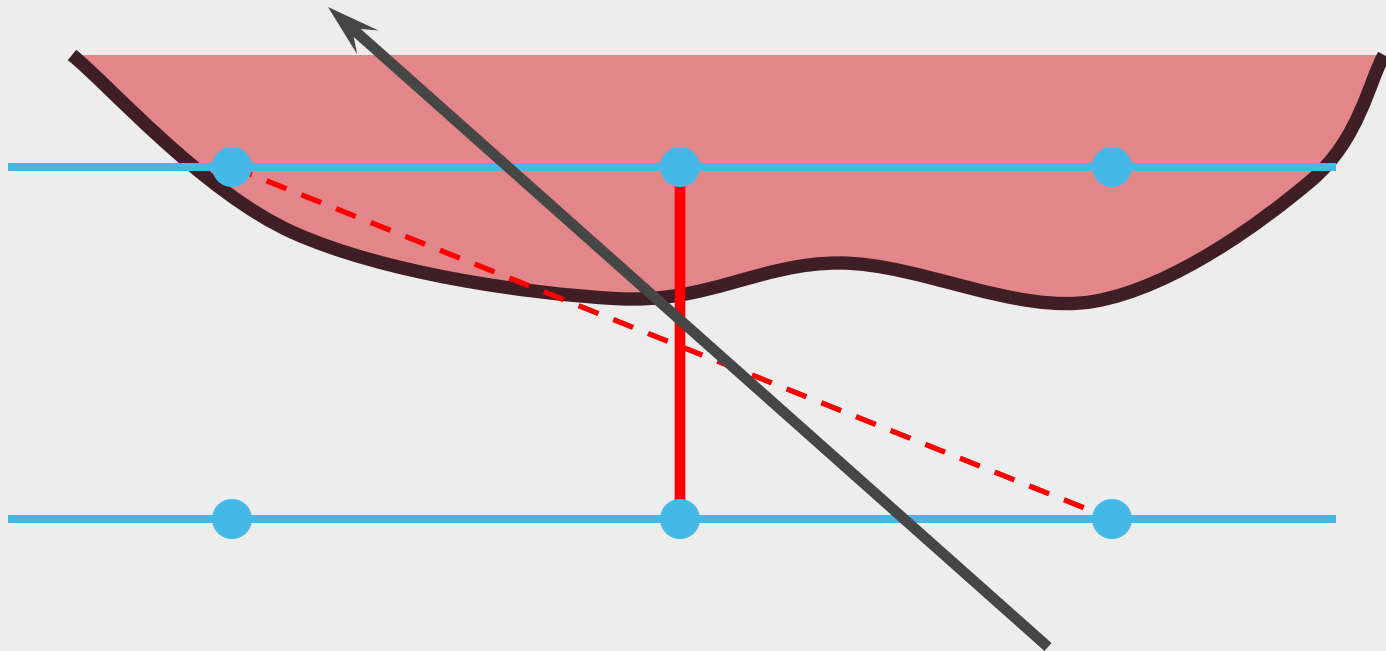
Lumigraph Rendering

- Use rough depth information to improve rendering quality



Lumigraph Rendering (cont.)

- Use rough depth information to improve rendering quality



Depth Correction

QuadralinearDepthCorrect(s,t,u,v,z)

Result = 0

$h_{st} = s_1 - s_0$ /* grid spacing */

$h_{uv} = u_1 - u_0$

for each of the four (s_i, t_j) surrounding (s, t)

$u' = u + (s - s_i) * z / (1 - z)$

$v' = v + (t - t_j) * z / (1 - z)$

temp = 0

for each of the four (u_p, v_q) surrounding (u', v')

interpWeight_{uv} =

$(h_{uv} - |u_p - u'|) * (h_{uv} - |v_q - v'|) / h_{uv}^2$

temp += interpWeight_{uv} * $L(s_i, t_j, u_p, v_q)$

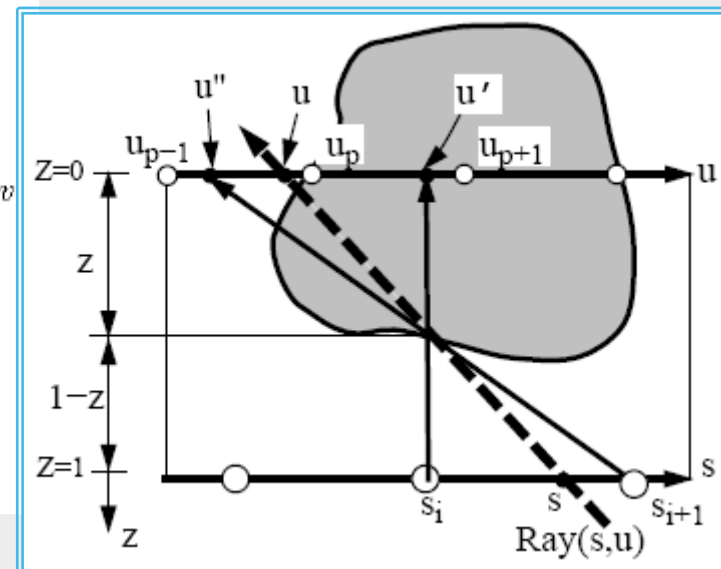
interpWeight_{st} =

$(h_{st} - |s_i - s|) * (h_{st} - |t_j - t|) / h_{st}^2$

Result += interpWeight_{st} * temp

return Result

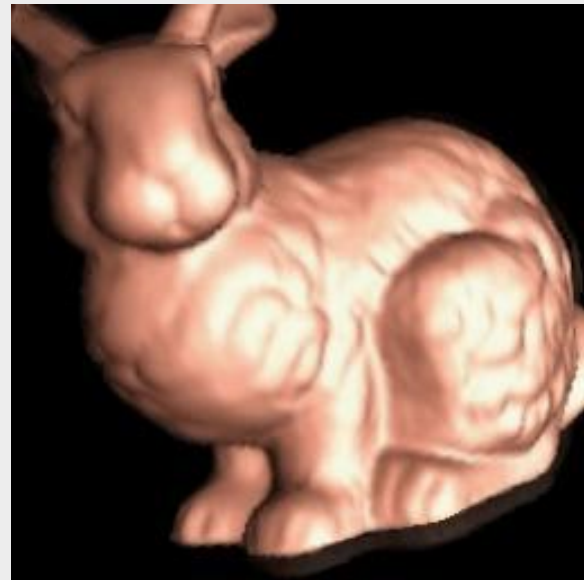
$$u' = u + (s - s_i) \frac{z}{1 - z}$$



Lumigraph Rendering (cont.)



Without using
geometry



Using approximate
geometry

Approximate Geometry

- Blue screen to find object in each image
 - Use octree to represent volume of object
 - Project, hierarchically, the octree cells
- Subdivide if not fully in/out.
- External polygons collected
 - Smoothed
- Using 4 levels of octree

