Outline

- **image-based rendering**
  - QuickTime VR
  - view interpolation
- **light fields**
  - vector versus scalar
  - 5D versus 4D
- **light field rendering**
  - parameterization
  - creation
  - display
  - digitization
  - compression
- **limitations**
- **geometry-image hybrids**
- **current research directions**

Interacting with 3D objects on a low-cost computer

Applications

- **web browsing**
- **online commerce**
  - » ***small*** **- merchandise catalogues**
  - » ***large*** **- real estate listings**
- **virtual museums**
- **video games**

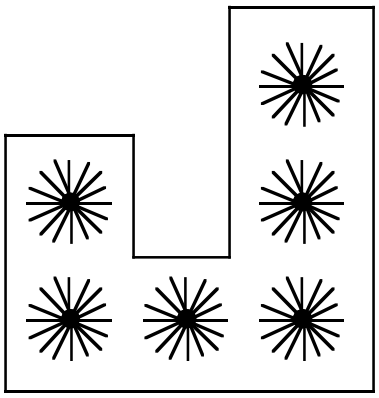
Technologies

- **canned video sequence**
- **rendered computer model**
- **image-based rendering**

Apple QuickTime VR

Outward-looking

- **panoramic views taken at regularly spaced points**



Inward-looking

- **views taken at points on the surface of a sphere**

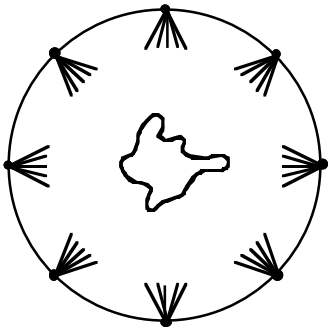


Image-based rendering

Flythroughs of 3D scenes from pre-acquired 2D images

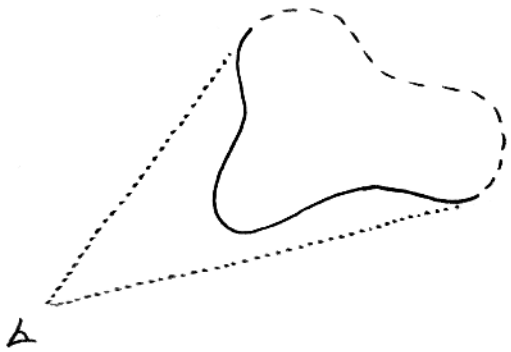
Advantages

- **low computation compared to classical CG**
- **cost independent of scene complexity**
- **imagery from real or virtual scenes**

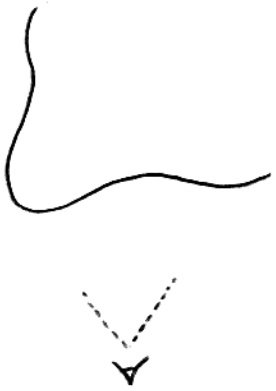
Limitations

- **static scene geometry**
- **fixed lighting**
- **fixed-look-from or look-at point**

View interpolation from a single view



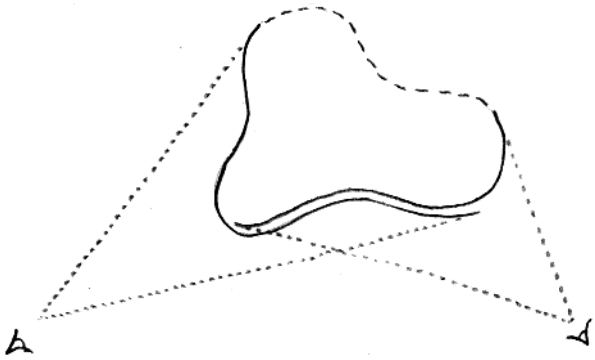
1. Render object
2. Convert Z-buffer to range image



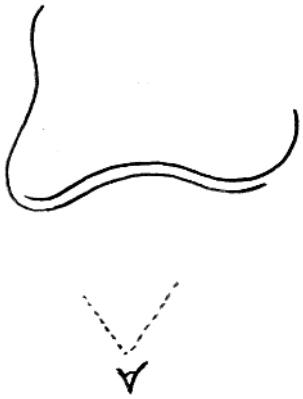
3. Re-render from new viewpoint
4. Use depths to resolve overlaps

Q. How to fill in holes?

View interpolation from multiple views



1. Render object from multiple viewpoints
2. Convert Z-buffers to range images



3. Re-render from new viewpoint
4. Use depths to resolve overlaps
5. Use multiple views to fill in holes

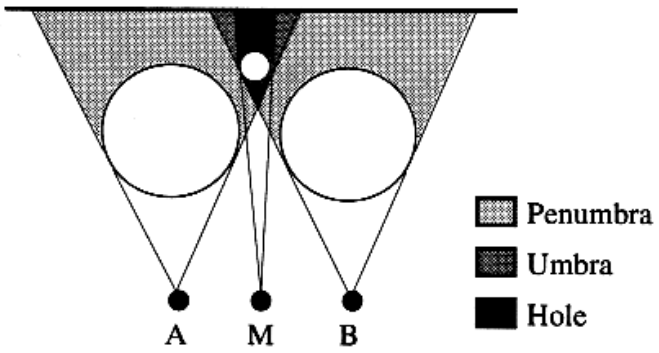
Problems with view interpolation

Resampling the range images

- block moves + image interpolation (Chen and Williams, 1993)
- splatting with space-variant kernels (McMillan and Bishop, 1995)
- fine-grain polygon mesh (McMillan et al., 1997)

Missed objects

- interpolate from available pixels
- use more views

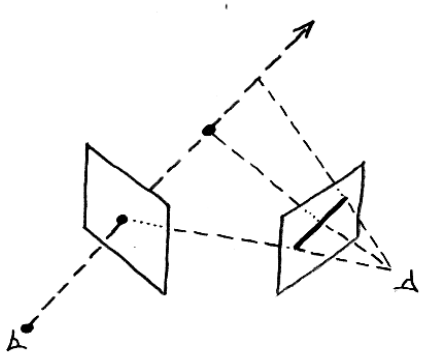


(from Chen and Williams)

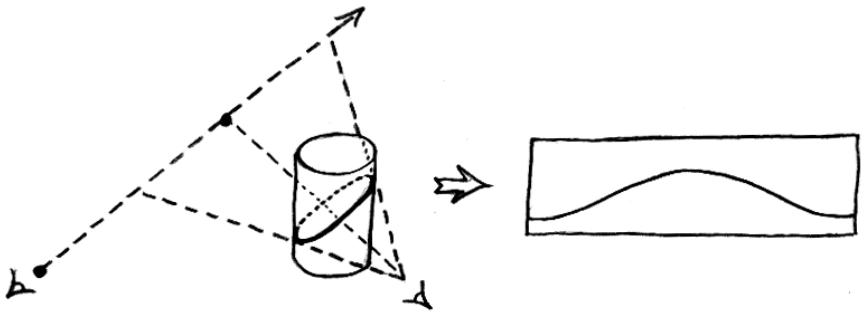
More problems with view interpolation

Obtaining range images is hard!

- use synthetic images (Chen and Williams, 1993)
- epipolar analysis (McMillan and Bishop, 1995)

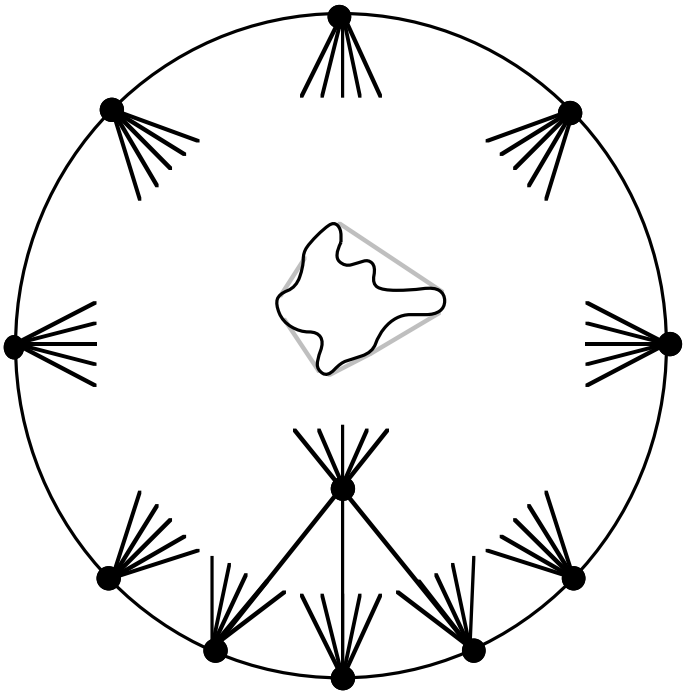


epipolar geometry



cylindrical epipolar geometry

A new solution: rebinning old views

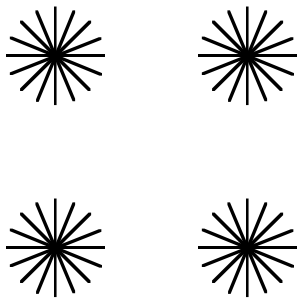


- must stay outside convex hull of the object
- like rebinning in computed tomography

Generalization: light fields

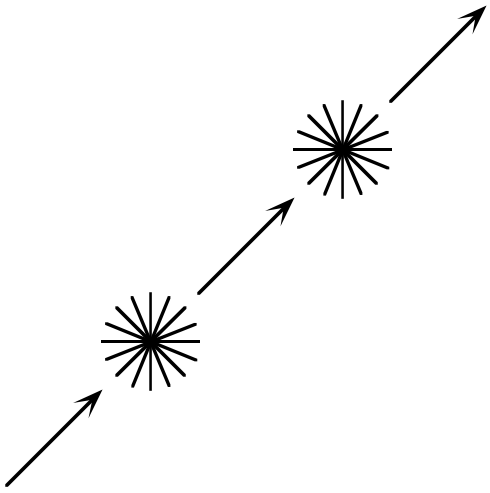
Radiance as a function of position and direction in a static scene with fixed illumination

- **For general scenes**
  $\Rightarrow$ 5D function



L (x, y, z, θ, ϕ)

- **In free space**
  $\Rightarrow$ 4D function



The vector light field

$$\Phi(\boldsymbol{x}) = \int_{S^2} \omega \; L(\boldsymbol{x}, \omega) \; d\omega$$

- **amplitude gives irradiance at point *x***
- **direction tells which way to orient a surface for maximum brightness**
- **applications to planning**
- **related concepts**
  - vector field (Gershun, 1936)
  - irradiance vector (Arvo, 1995)
  - photic (scalar) field (Moon, 1981)
  - plenoptic function (Adelson and Bergan, 1991)

The “free space” assumption

- flying around a small object
- flying through the interior of an environment

Using light fields

- images are 2D slices of the 4D function
- populate using any set of slices
- construct arbitrary views by taking other slices

Related work

- **epipolar volumes (Bolles, Baker, Marimont)**
- **holographic stereograms (Benton)**
- **near-field photometry (Ashdown)**

Key technical challenges

- **Parameterization**
- **Creation**
- **Display**
- **Digitization**
- **Compression**

Parameterization issues

---

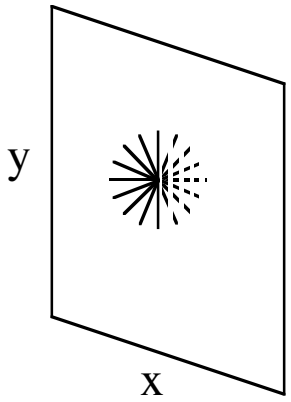
1. **Efficient calculation**

2. **Controlled coverage**

3. **Uniform sampling**

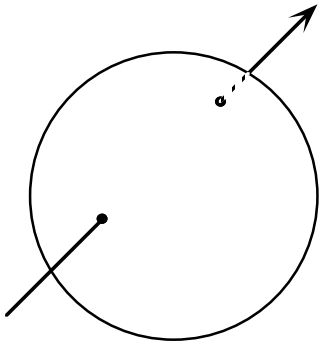
Some candidate parameterizations

Point-on-plane + direction



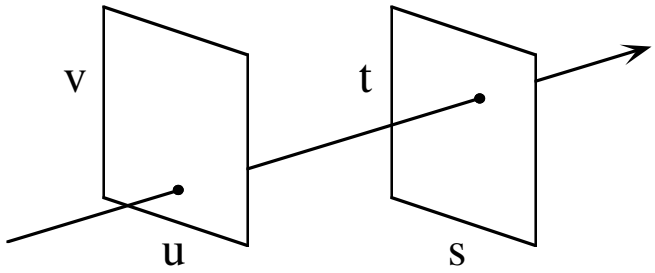
$$L (x, y, \theta, \phi)$$

Chords of a sphere



$$L (\theta_1, \phi_1, \theta_2, \phi_2)$$

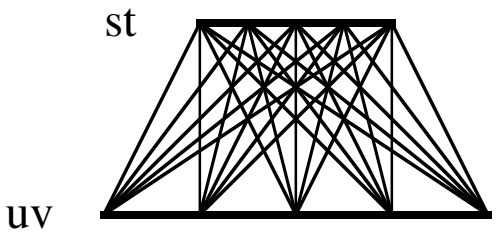
Two planes ("light slab")

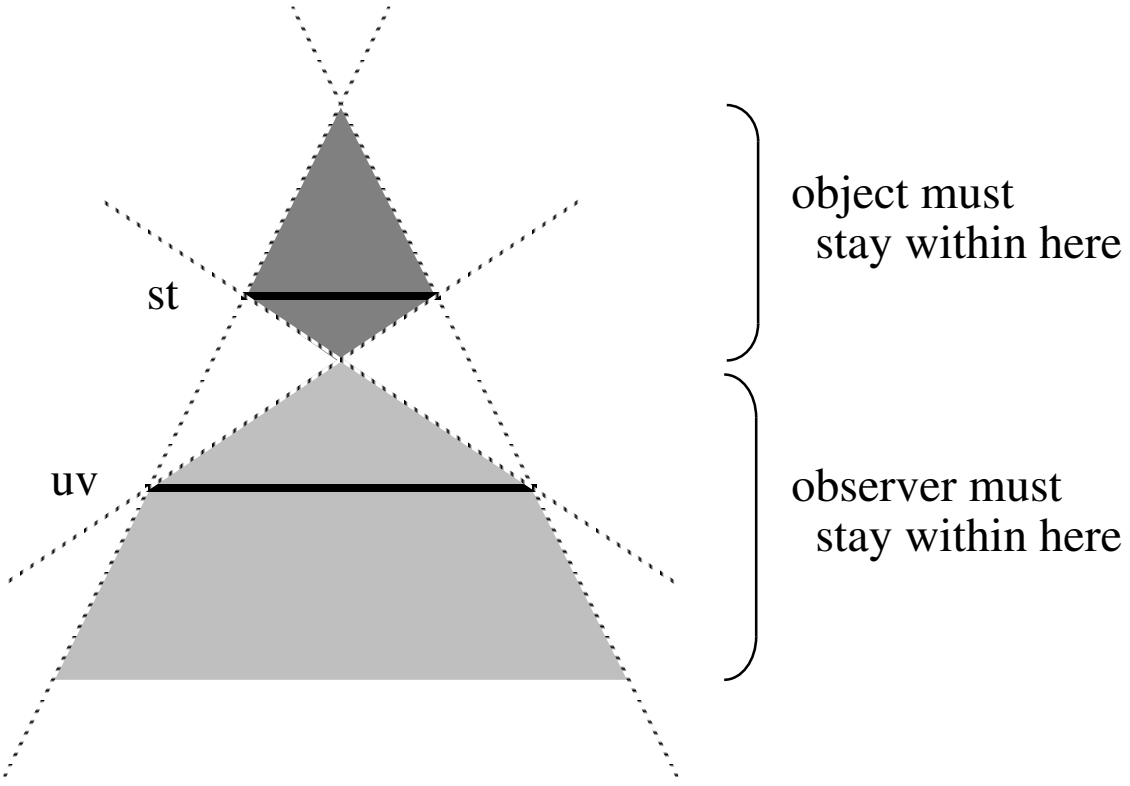


L (u, v, s, t)

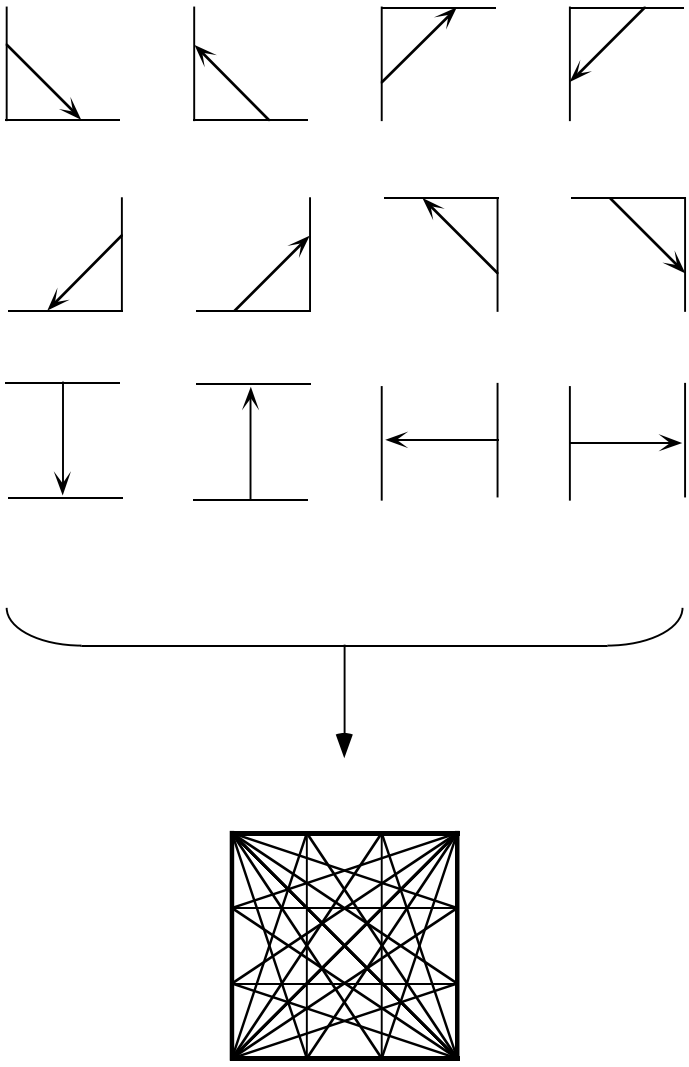
- **planes in arbitrary position**
- **uses projective geometry**
- **fast incremental algorithms**

The coverage of a light field



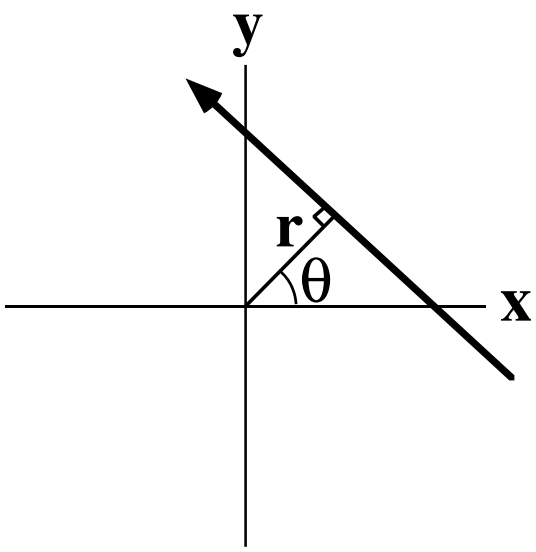


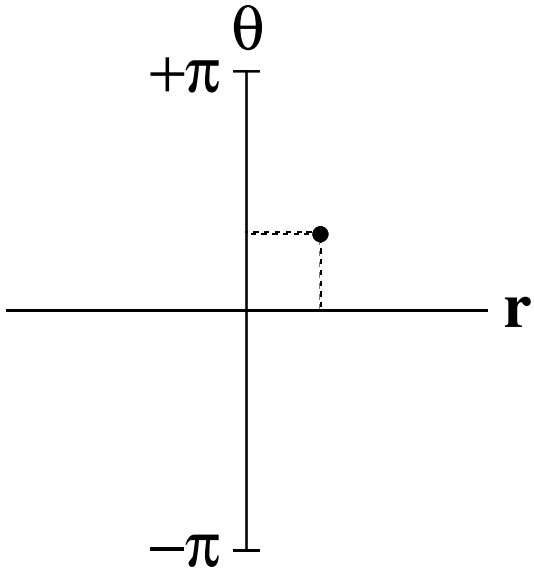
An omni-directional light field



- **object can fill square**
- **observer can stand anywhere outside square**

Line space interpretation





point space

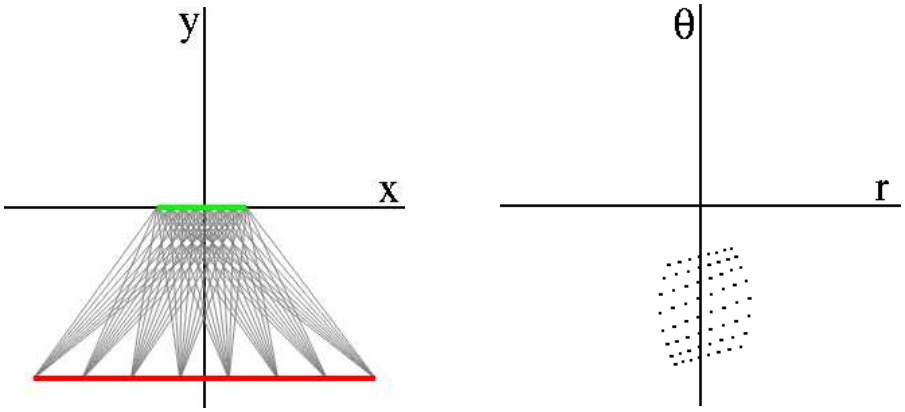
line space

Using line space to visualize ray coverage

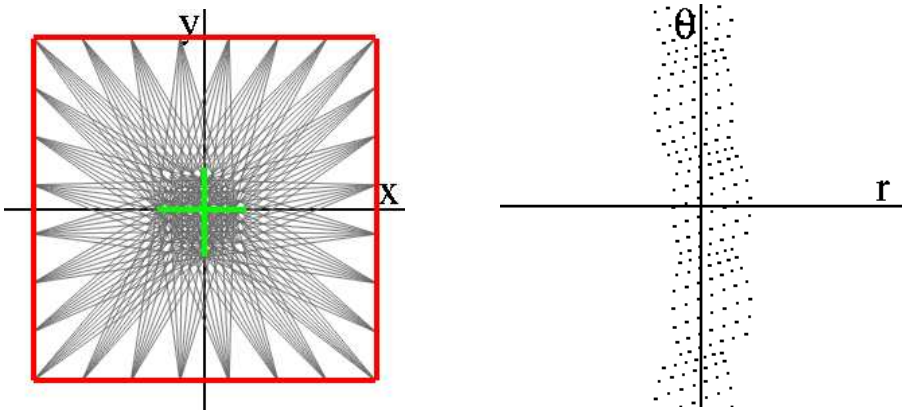
---

point space **line space**

one slab



four slabs

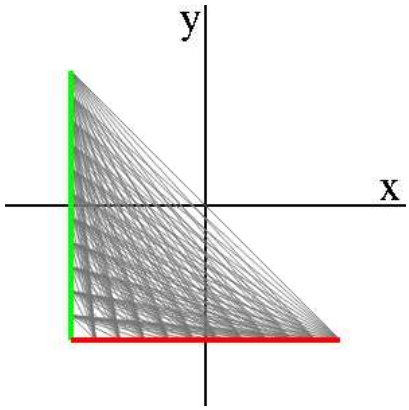


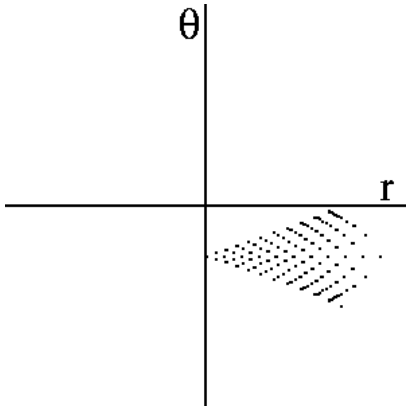
Using line space to visualize sampling uniformity

---

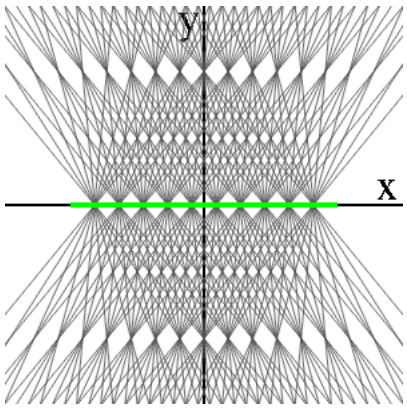
point space **line space**

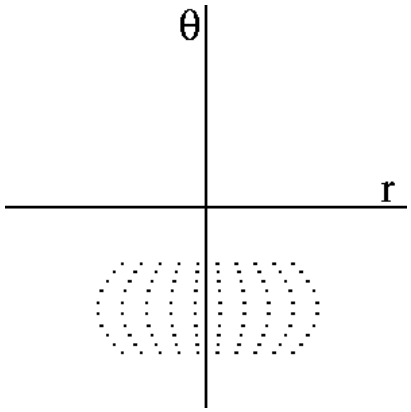
corner





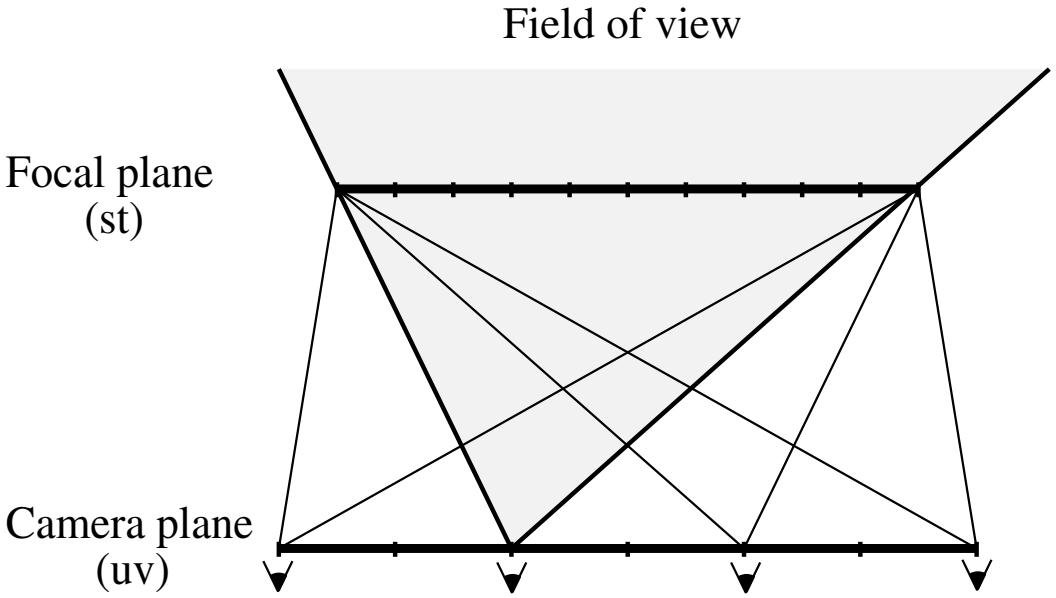
orthographic





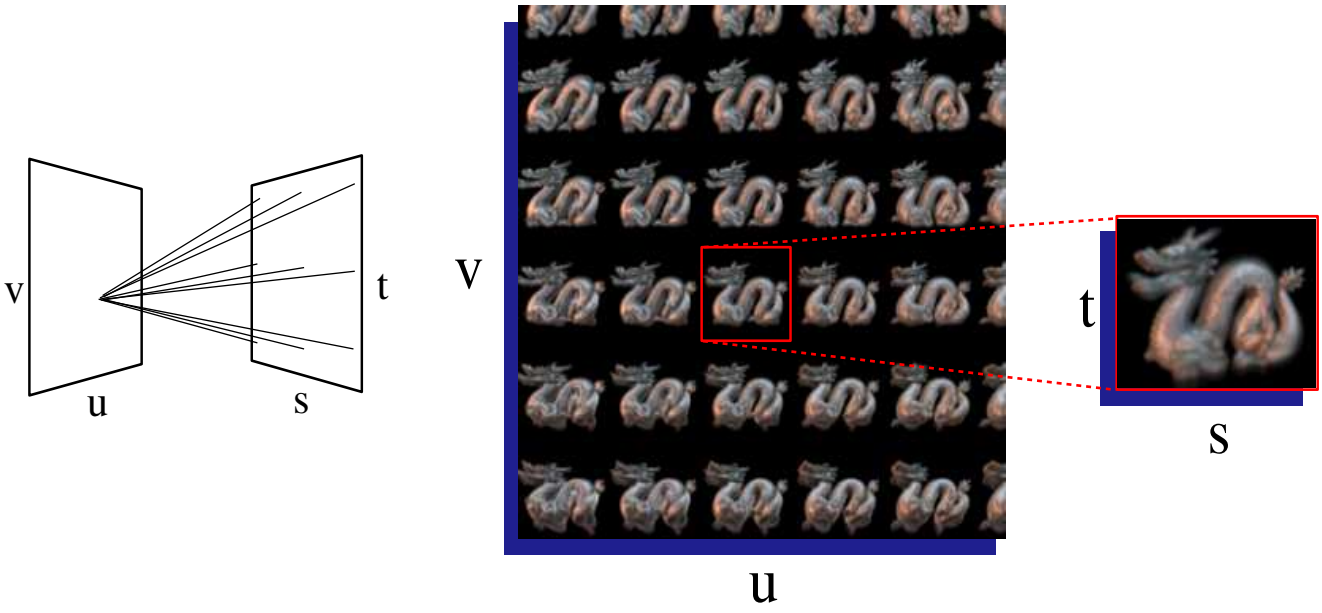
- **dense sampling in point space ⇔ dark regions in line space**

Creating a light field

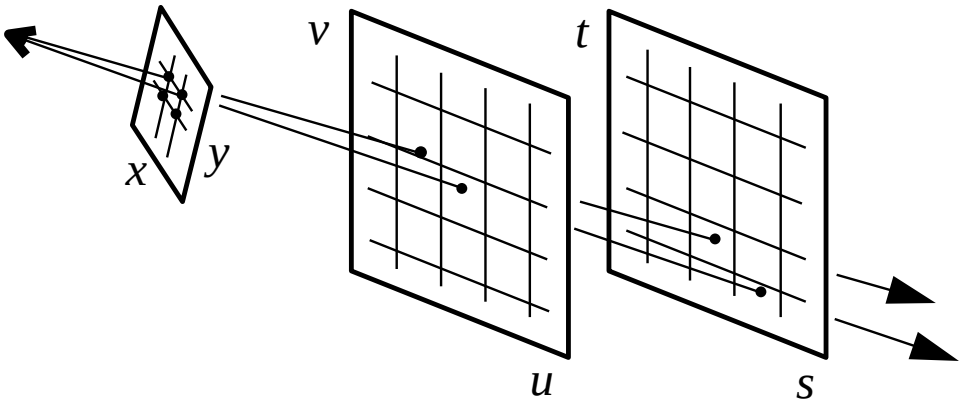


- **off-axis perspective views**

A light field is an array of images

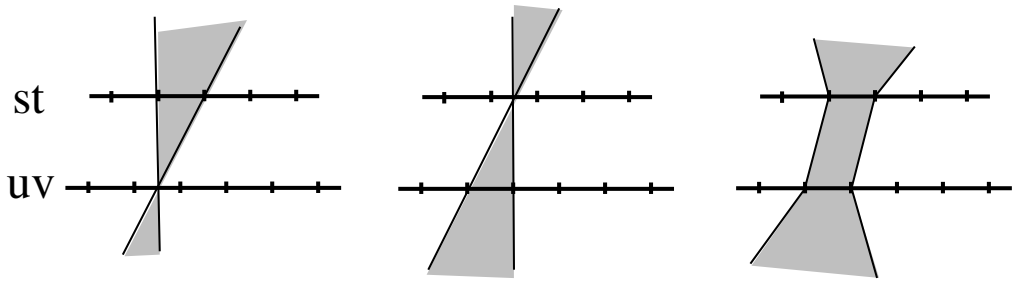


Displaying a light field

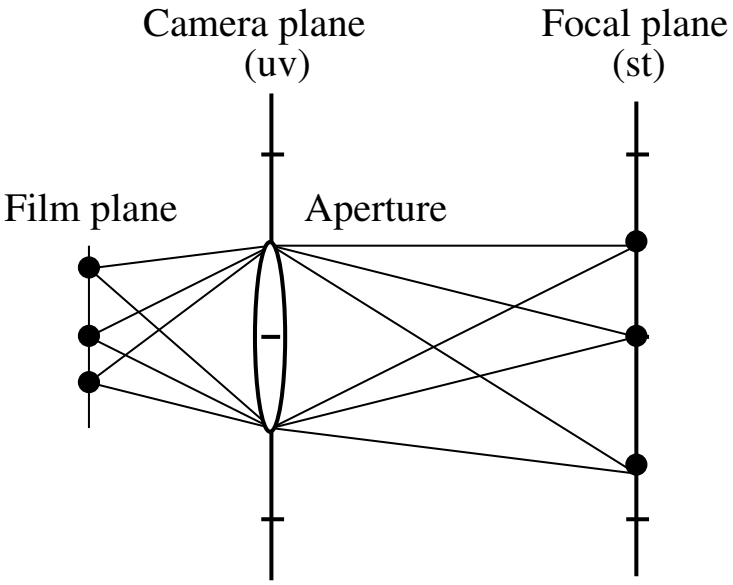



```
foreach x,y
    compute u,v,s,t
    I(x,y) = L(u,v,s,t)
```

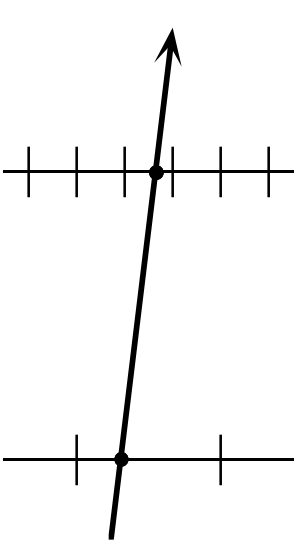
Prefiltering during creation



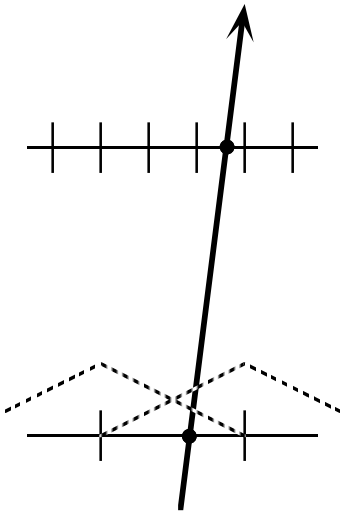
Pixel filter + Aperture filter = Ray filter



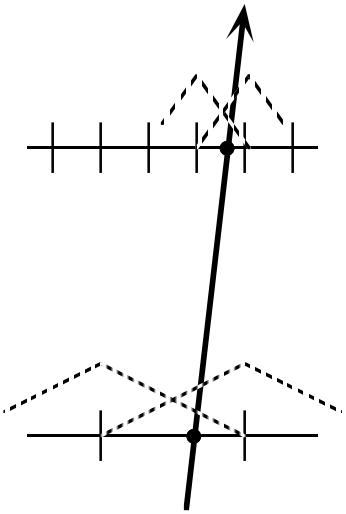
Interpolation for display



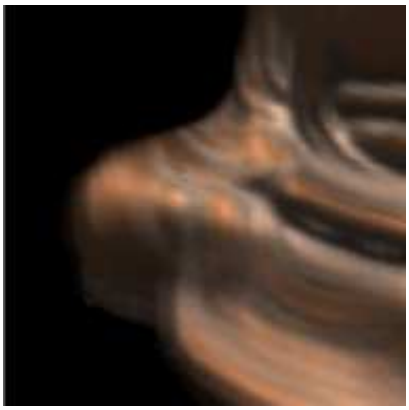
point sample



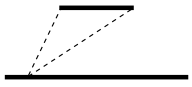
uv bilerp



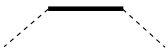
uv and st bilerp



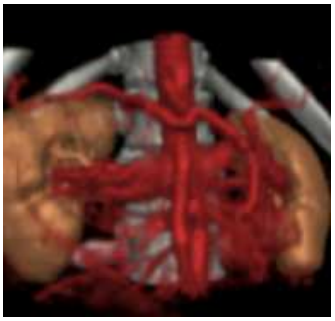
Examples

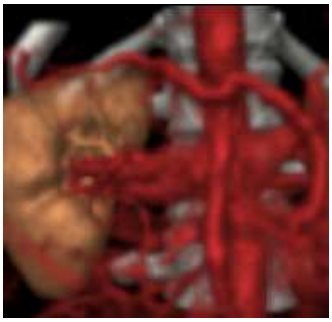


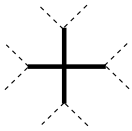
buddha



kidney







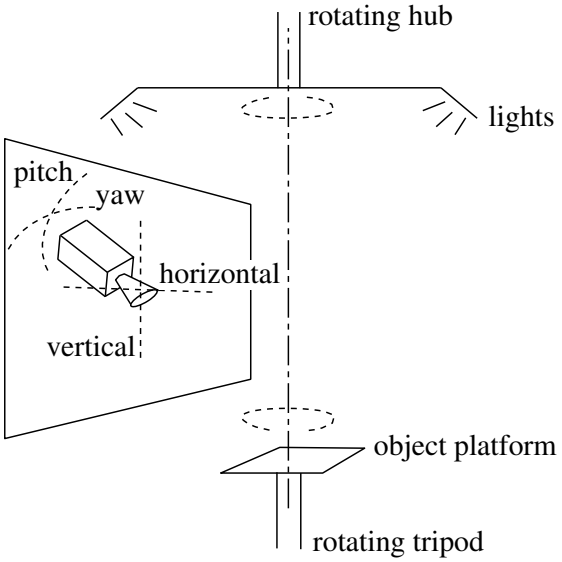
hallway

lion

Digitization issues

- **inward versus outward looking**
- **human versus computer controlled**
- **spherical versus planar camera motion**

A prototype gantry



More issues

- location of focal plane
- aperture antialiasing
- object support
- lighting

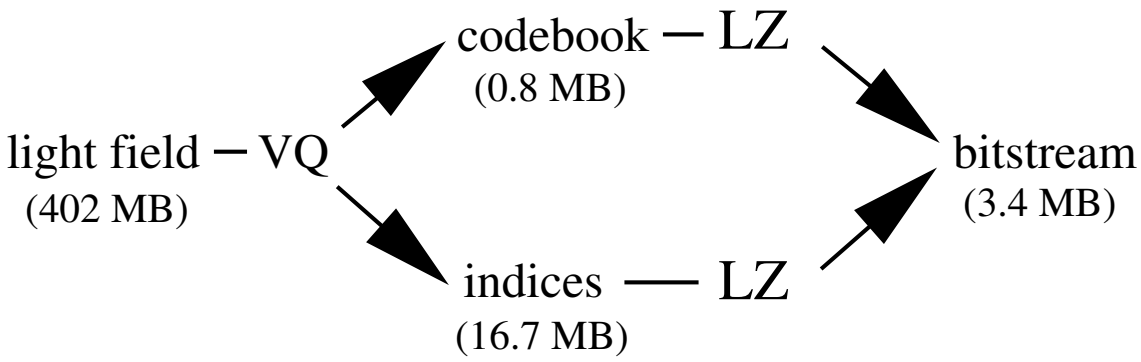
Hand-held camera (Gortler et al., 1996)

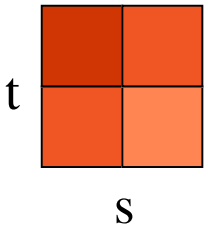
- **Camera calibration (intrinsics) & pose estimation (extrinsics)**
  - background scene with reference markers
  - algorithm of Tsai, 1987
- **Scattered data interpolation**
  - splat-pull-push

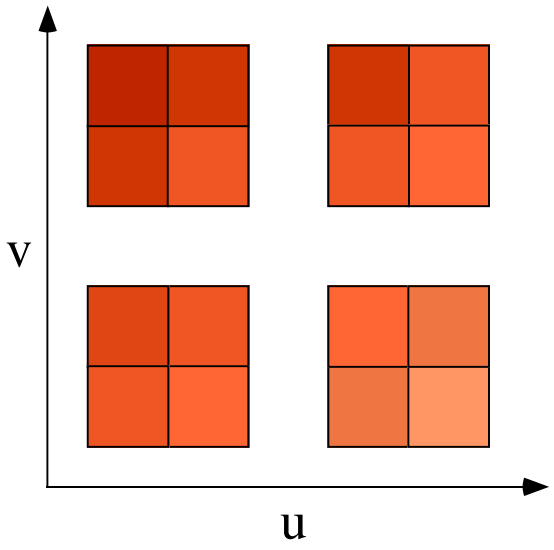
Compression issues

- **need ≥ 100:1**
- **4D redundancy**
- **random access to samples**
- **fast decoding**
- **offline encoding**

Vector quantization + entropy coding







2D tile

4D tile

Compression example

---

original

compressed 120:1

Performance summary for full-surround lion

Digitization

– number of slabs	4
– images per slab	32 x 16
– pixels per image	256 x 256
– raw size	402 MB

Compression

– VQ coding	24 : 1
– entropy coding	5 : 1
– total compression	120 : 1
– compressed size	3.4 MB

Timings

– acquisition time	5 hours
– compression time	4 hours
– update rate	6 frames / sec

Limitations

Closely spaced views

- **dense sampling → redundancy→ good compression**

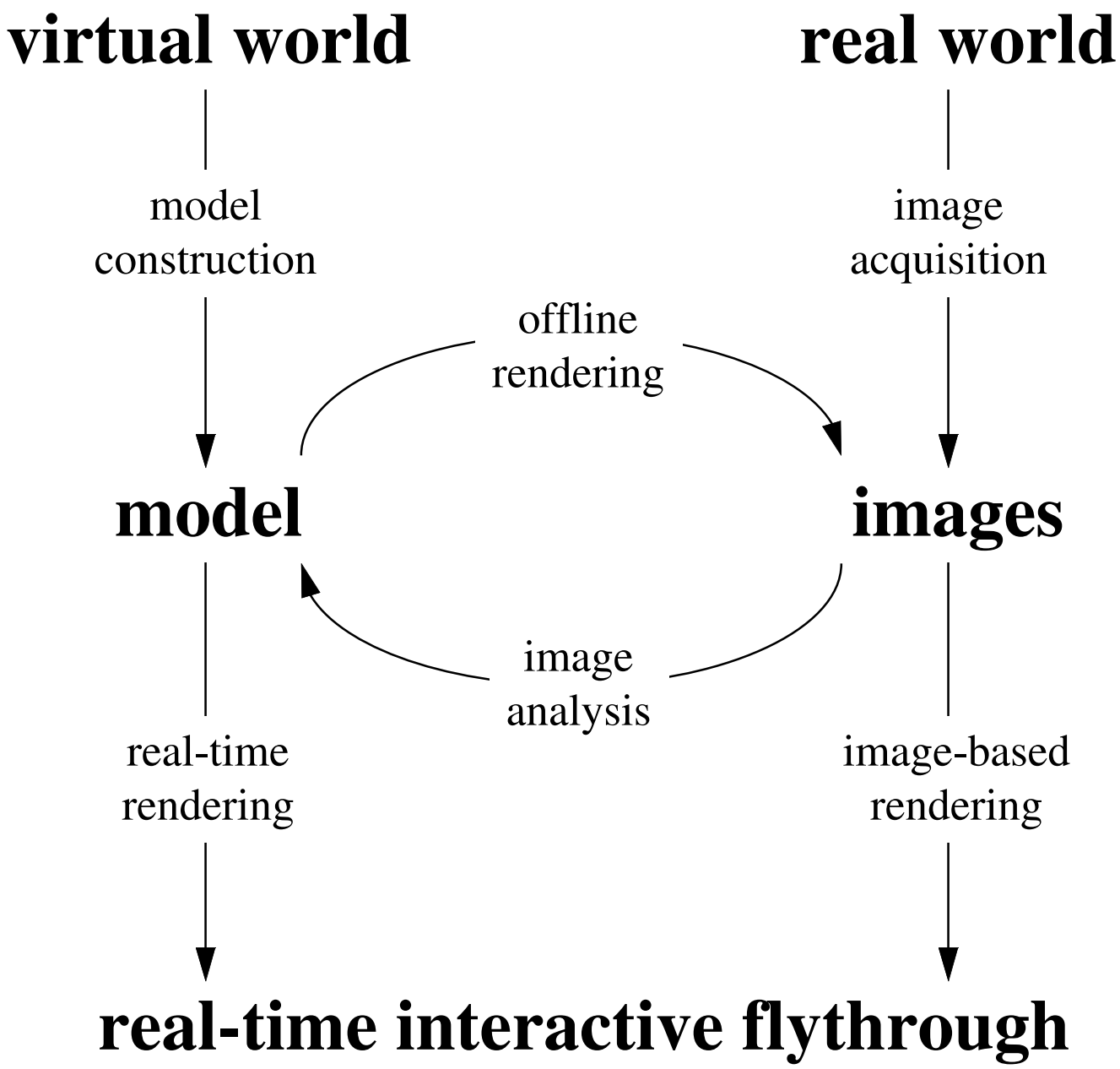
Free-space restriction

- **stitch together multiple light fields**
- **light fields + Z-buffer**

Fixed illumination

- **4D lights × 4D views = 8D**
- **light fields + normals and optical properties**
- **superposition of illumination basis functions**

Model-based versus image-based rendering



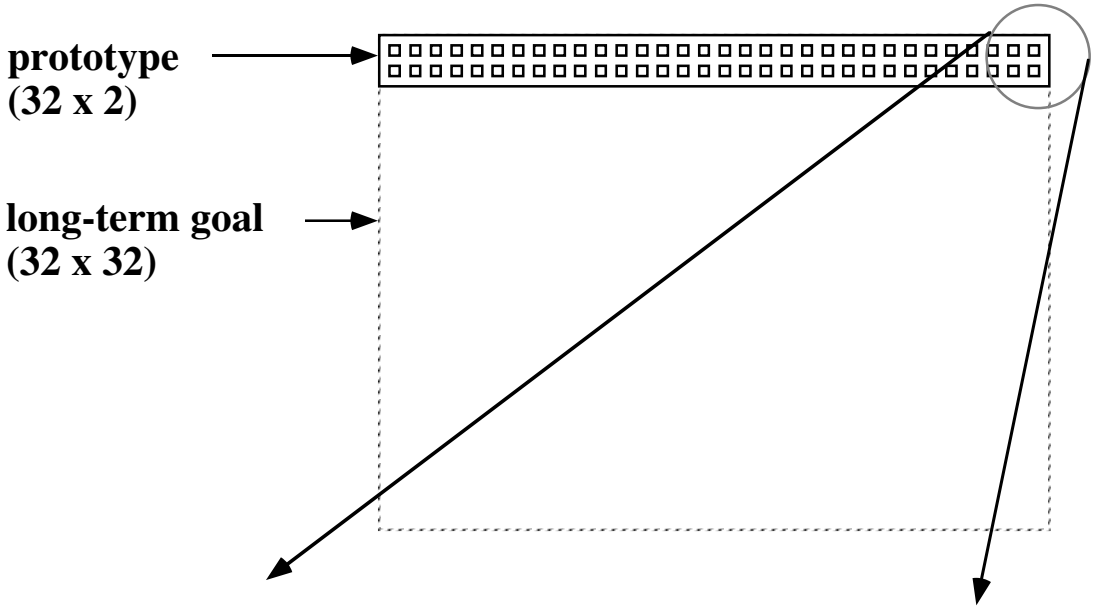
Hybrid geometry / light field representations

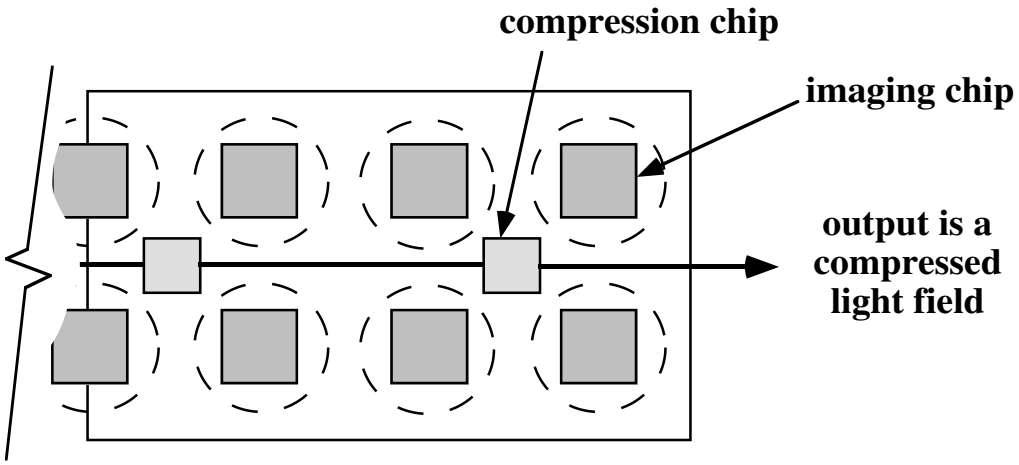
- **depth-corrected LF rendering**
  - depth-from-silhouettes -> rough 3D model -> corrected basis functions for reconstruction (Gortler et al., 1996)
- **geometry-assisted LF compression**
  - 4D subdivision -> shape-from-light-fields -> polygon + texture + 4D residual (Pereira and Zhu)
- **object-centered LF representation**
  - radiance function per vertex on fine mesh (Levoy)
  - adaptive ray tracing -> light field per object w. 4D subdivision -> 4D interpolation (Teller et al., 1996)

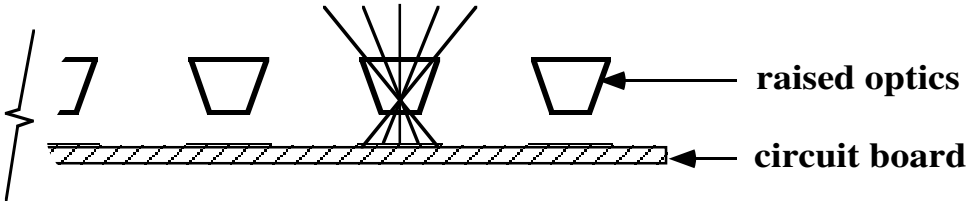
Future directions

- **Authoring and viewing software for light fields**
  - light **fields as first-class primitives**
- **Light field caching for interactive flythroughs**
  - hierarchical light fields
- **Real-time digitization**
  - arrays of cameras
- **Computer vision algorithms**
  - shape-from-light-fields
- **Measuring and representing reflectance**
  - converting light fields (radiance) to BRDFs (reflectance)

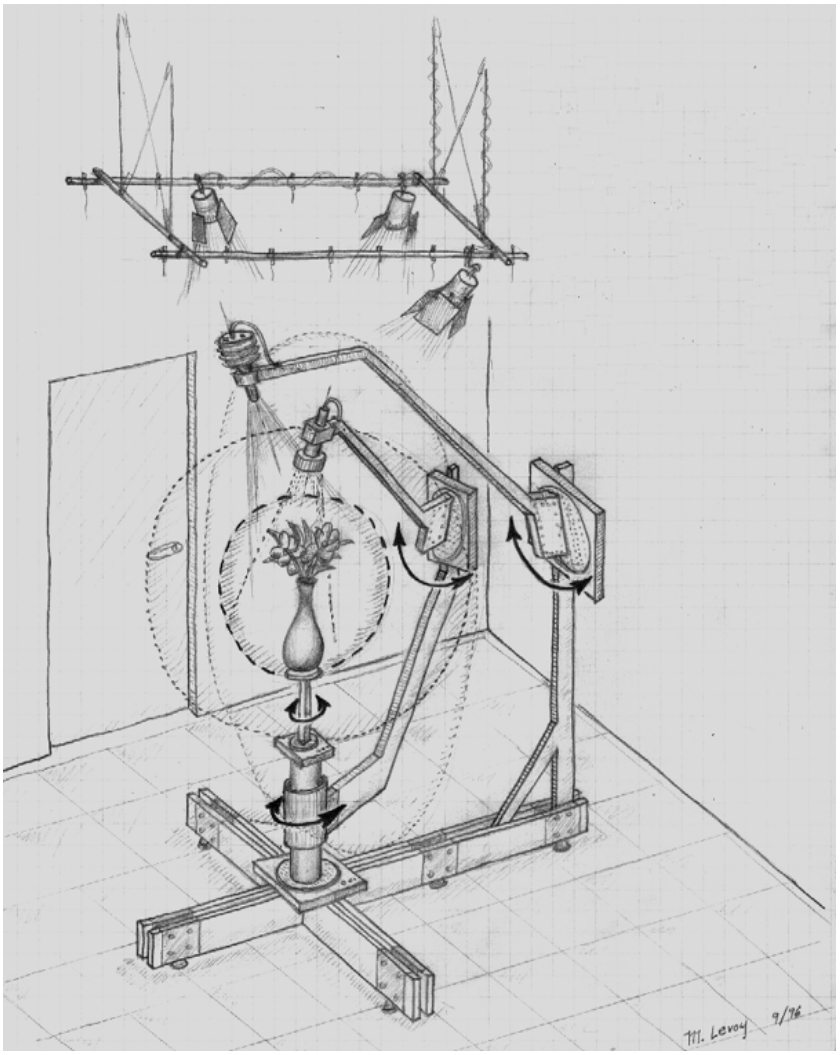
Planar array of cameras







Multi-purpose camera gantry



- **digitizing light fields**
- **measuring BRDFs**
- **range scanning**