Image-based rendering

"The study of image-based modeling and rendering is the study of sampled representations of geometry."

- classic representations
  - textures, sprites, shadow maps, reflectance maps, range images, disparity maps, optic flow fields, panoramas, movie maps, density volumes, epipolar volumes, etc.

- recent additions
  - arrays of images, light fields / Lumigraphs, inconsistent arrays of range images, hierarchical image caches, etc.

Taxonomy of IBR techniques

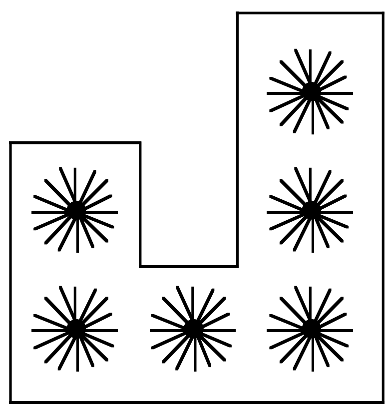
- no model
  - environment map
  - QuickTime VR (QTVR)
- camera pose
  - sprite compositing
  - light fields
- image correspondences
  - plenoptic editing
- inconsistent range images
  - view interpolation
- full model
  - volume rendering

more knowledge of scene geometry

Apple QuickTime VR

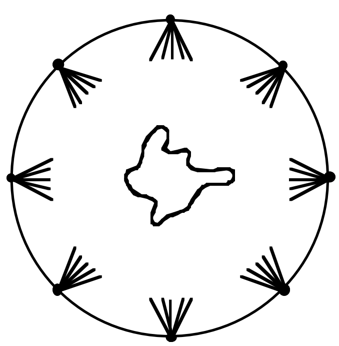
Outward-looking

– panoramic views taken at regularly spaced points

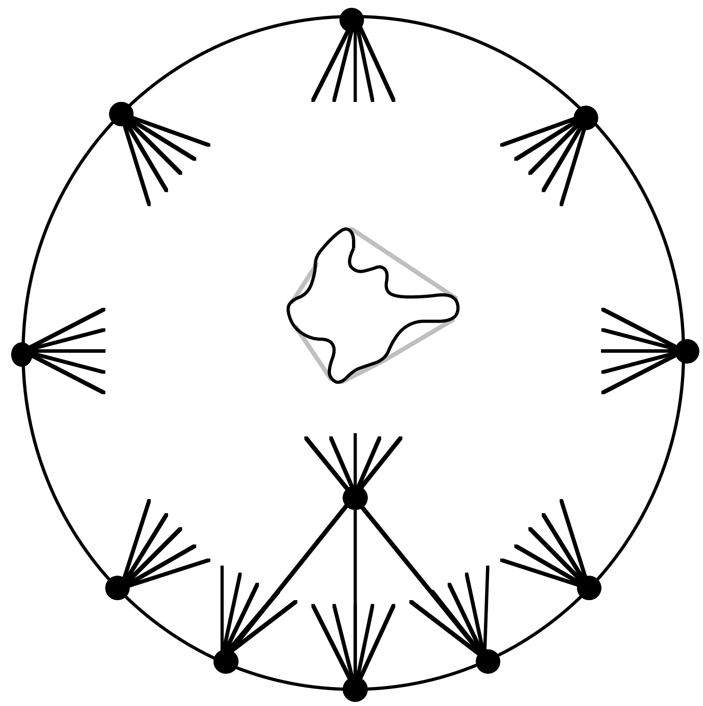


Inward-looking

– views taken at points on the surface of a sphere



Light field rendering



Key ideas

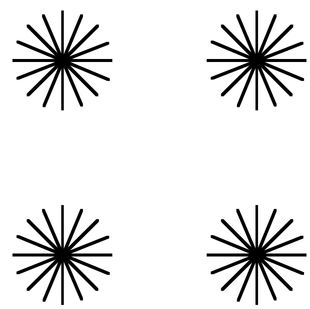
- construct new views by rebinning old views
- robust because it never builds a 3D model
- must stay outside the convex hull of the object
- fixed scene geometry and lighting

The scalar light field

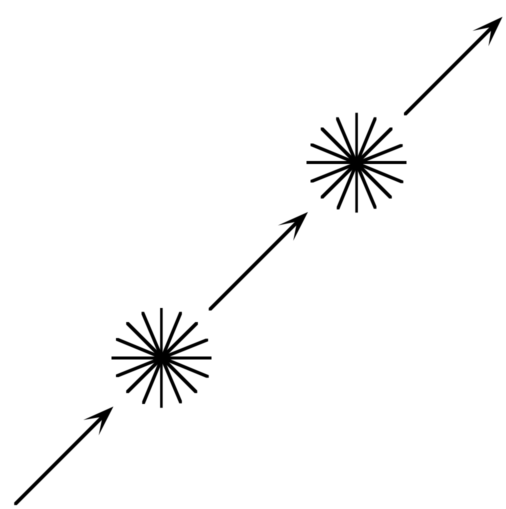
Radiance as a function of position and direction in a static scene with fixed illumination

For general scenes
$\Rightarrow$ 5D function

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



In free space
$\Rightarrow$ 4D function



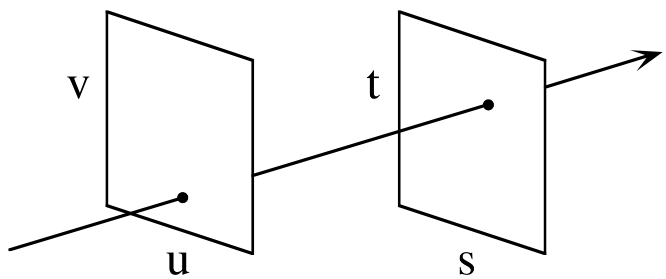
The vector light field

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

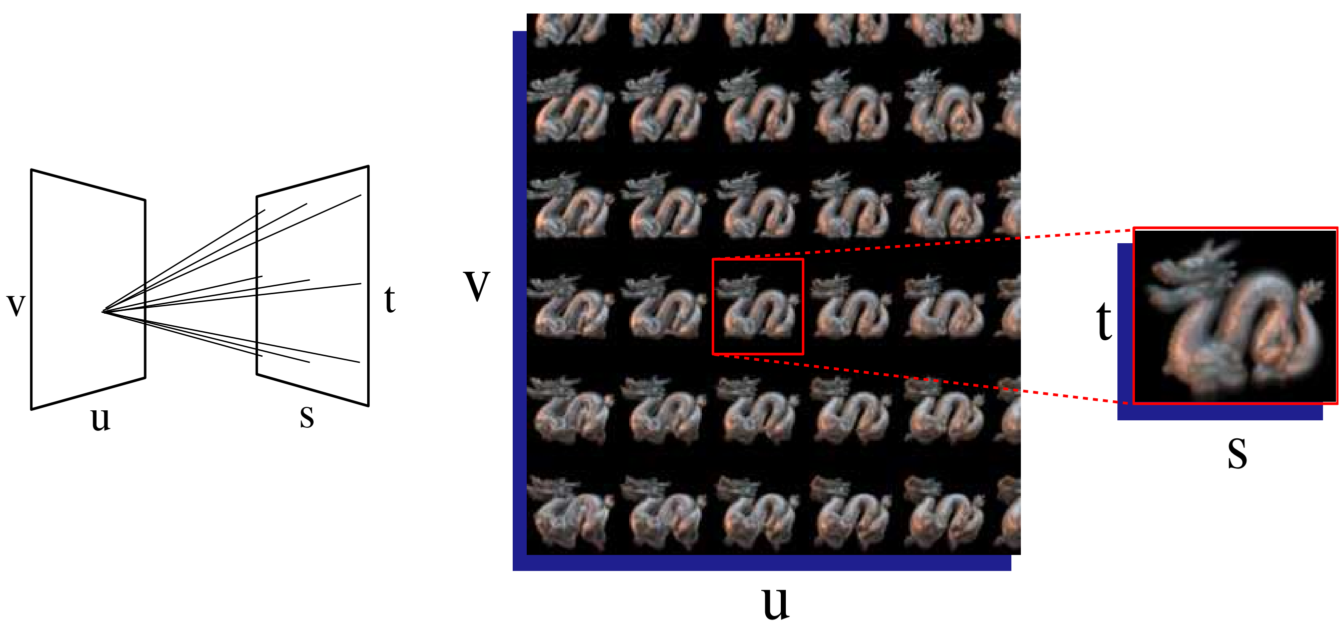
- amplitude gives irradiance at point *x*
- direction tells which way to orient a surface for maximum brightness
- history
  - vector field (Gershun, 1936)
  - photic (scalar) field (Moon, 1981)
  - plenoptic function (Adelson and Bergen, 1991)
  - irradiance vectors and tensors (Arvo, 1995)

A practical light field rendering algorithm

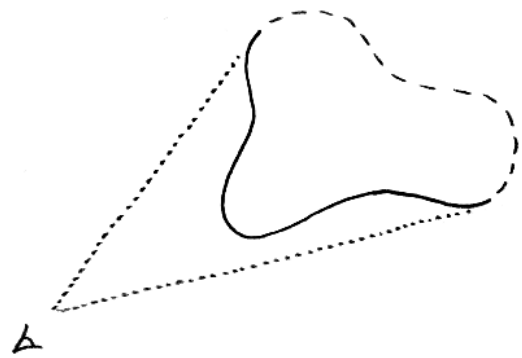
Two-plane parameterization



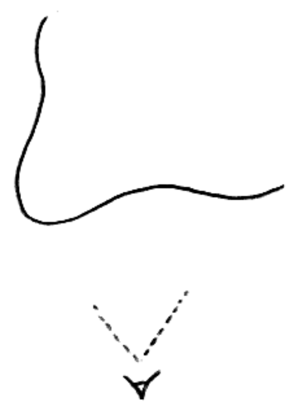
A light field is an array of images



View interpolation from a single view



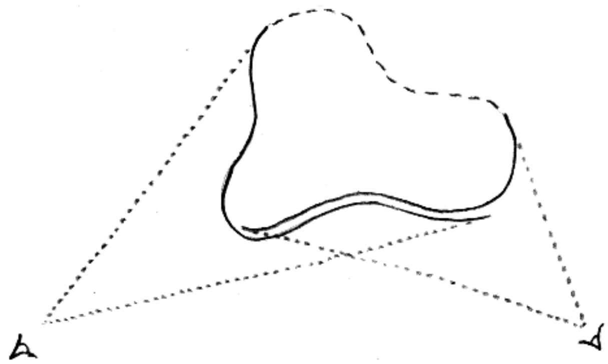
1. Render object
2. Convert Z-buffer to range image



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

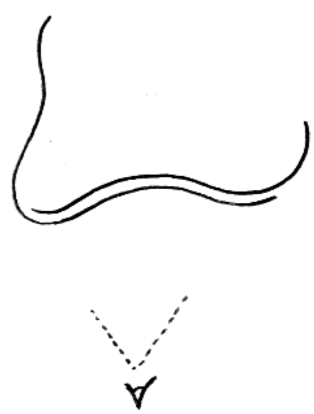
Q. How to fill in holes?

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

Future prospects for IBR

Advantages

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

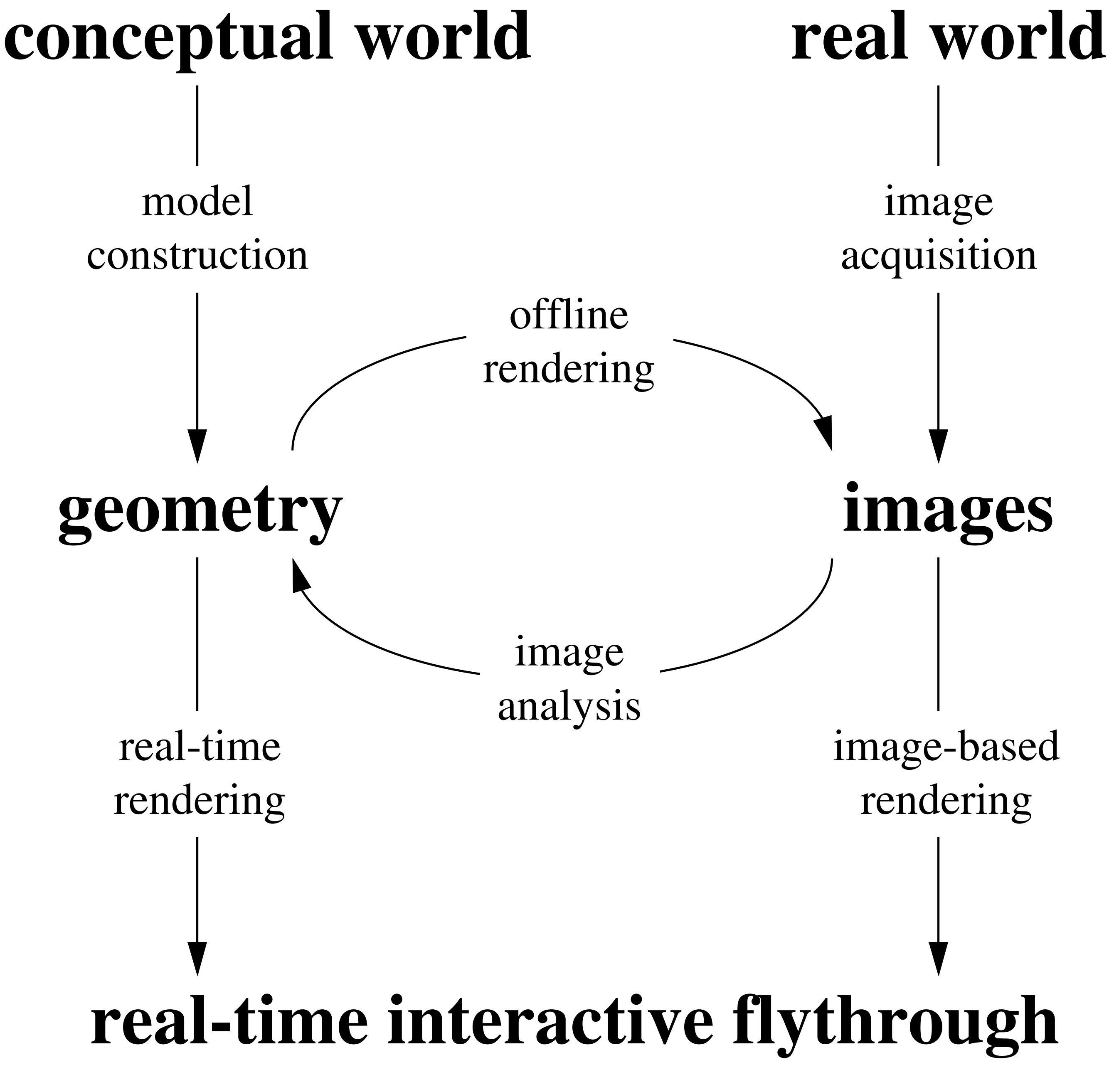
Limitations & solutions

– closely spaced views	improved compression
– free-space restriction	hierarchical light fields
– fixed lighting	illumination basis fncns
– static scene geometry	rerendering

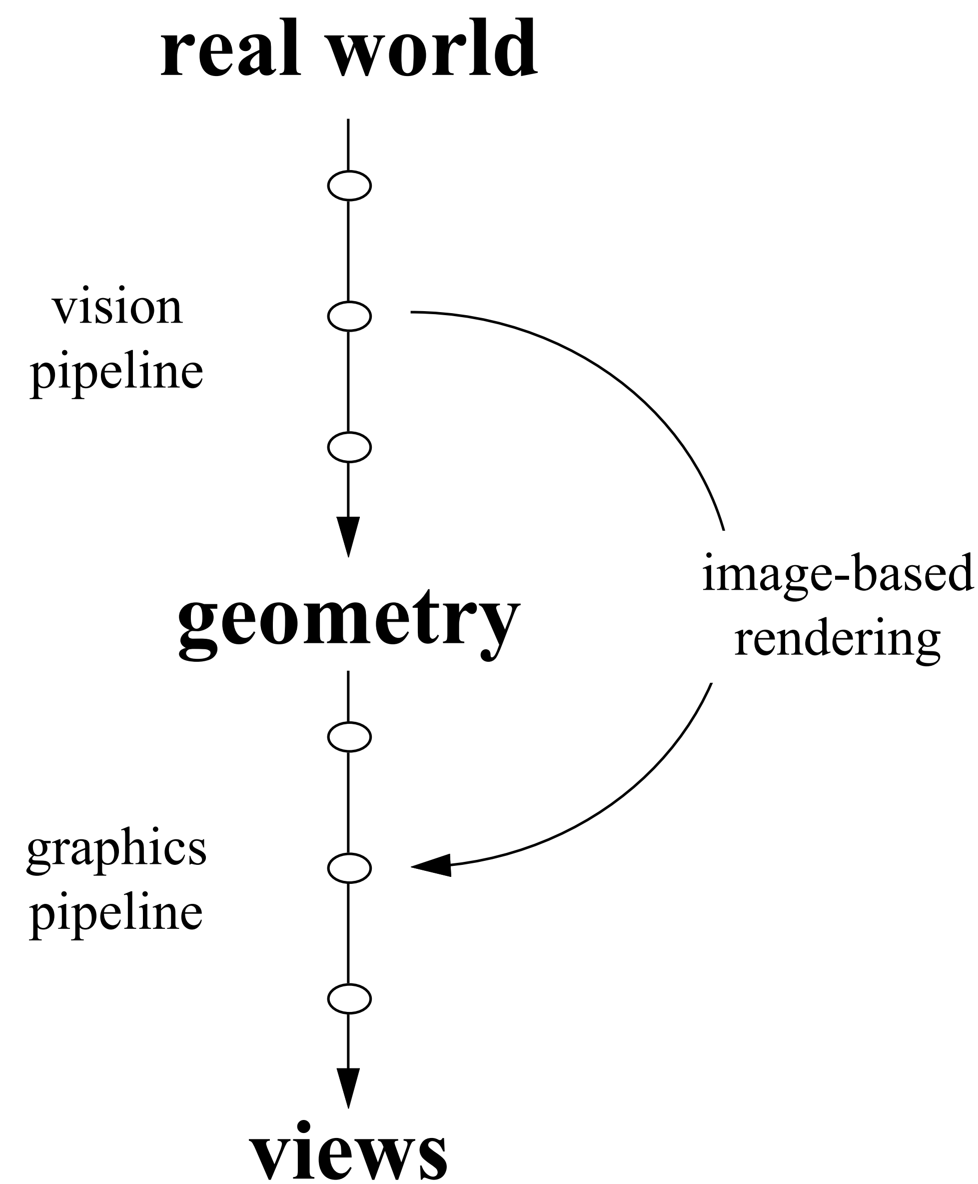
Related research (in our lab)

– view interpolation + morphing
– devices for acquiring light fields
– measuring and representing reflectance
– shape-from-light-fields

Geometry-based versus image-based rendering



Shortcutting the vision/graphics pipeline



(from M. Cohen)