Acquisition and Rendering of Transparent and Refractive Objects

Wojciech Matusik[1], Hanspeter Pfister[2],
Remo Ziegler[2], Addy Ngan[1], Leonard McMillan[1].

1. MIT - wojciech, addy, mcmillan@graphics.lcs.mit.edu
2. MERL - pfister, ziegler@merl.com

EGRW 02

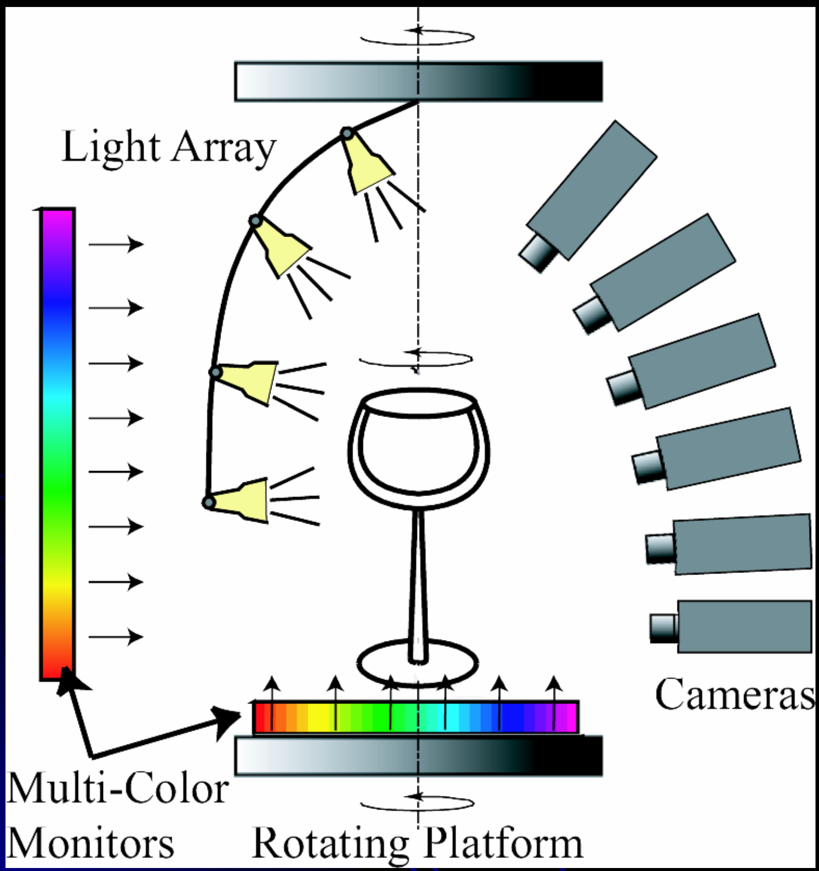
Overview

- An image-based 3D scanning system
  - Handle refractive, transparent objects
  - Can be rendered in novel environments
  - Robust, automatic
  - Approximate geometry based on *visual hull*
  - View-dependent opacity and texture

Previous Works

- Active and passive 3D scanners
  - Work best for diffuse materials
  - Fuzzy, transparent and refractive objects are difficult
- Reflectance field [Debevec et al 00]
  - Light Stage system to capture reflectance fields
  - Fixed viewpoint
- Environment matting [Zongker et al 99, Chuang et al 00]
  - Extension to allow for reflections and refractions
  - Fixed viewpoint

The System

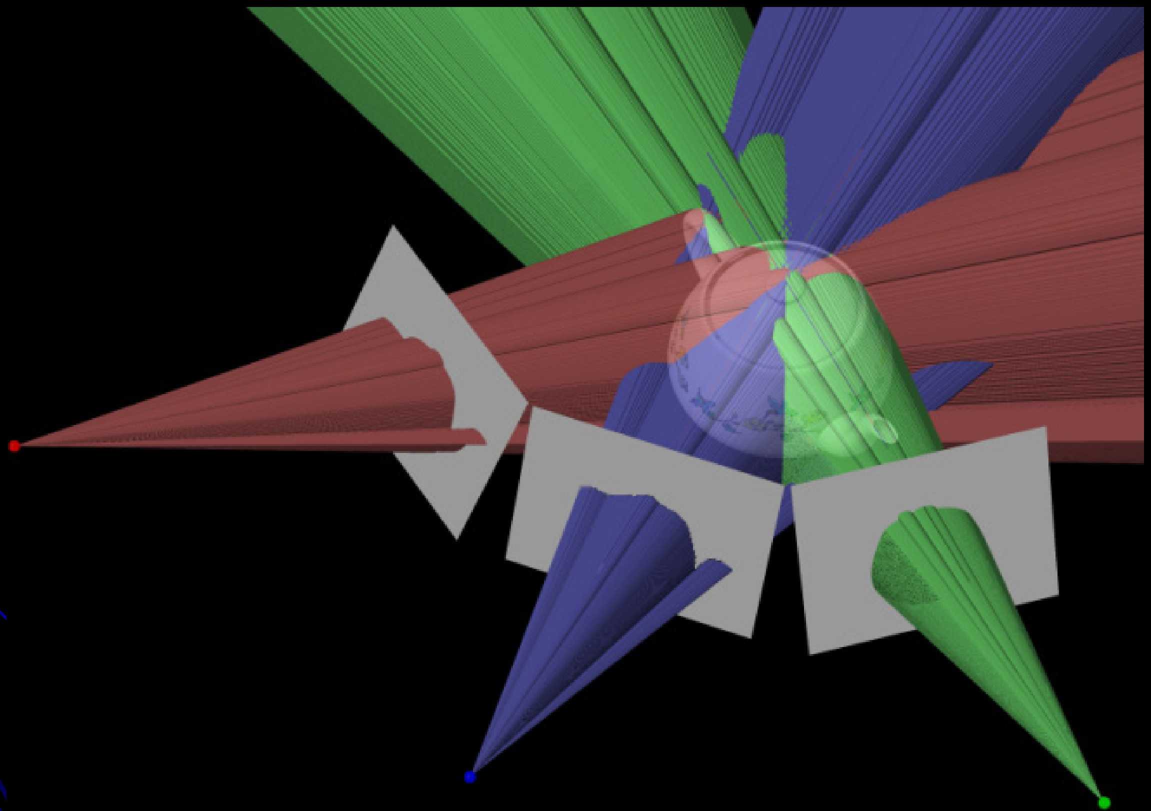

Light Array
Cameras
Multi-Color
Monitors
Rotating Platform

Acquisition

- For each viewpoint (6 cameras x 72 positions)
  - Alpha mattes
    - Use multiple backgrounds [Smith and Blinn 96]
  - Reflectance images
    - Pictures of the object under different lighting (4 lights x 11 positions)
  - Environment mattes
    - Use similar techniques as [Chuang et al. 2000]

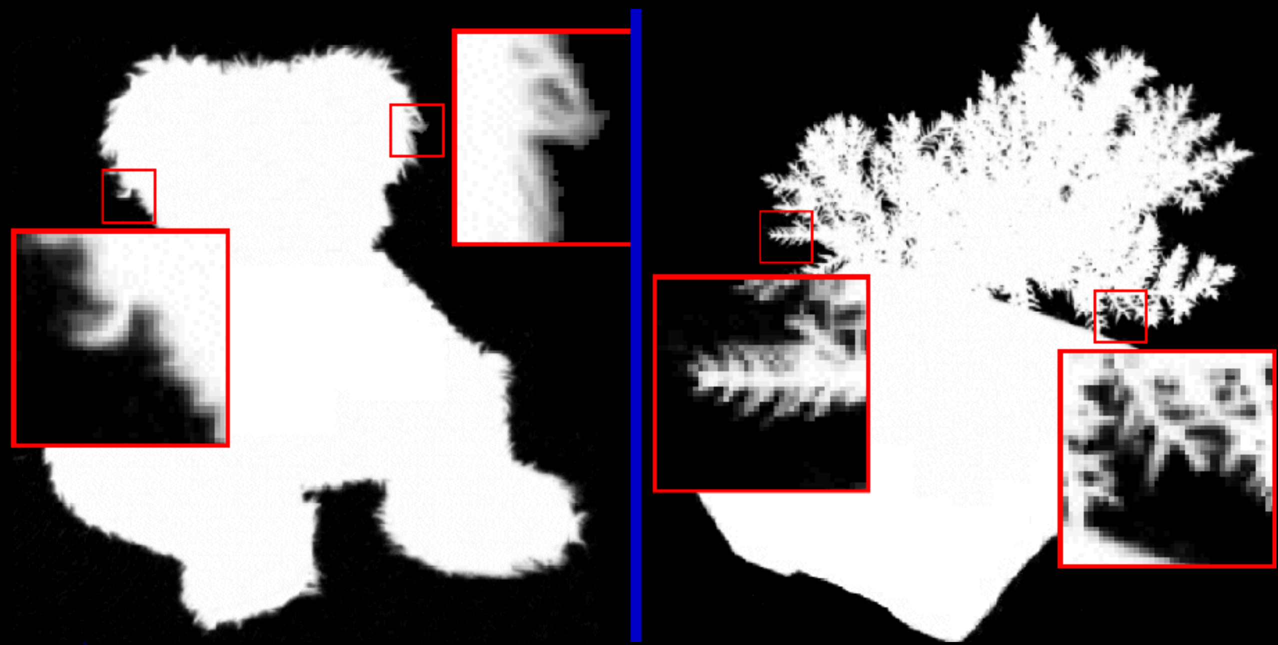
Geometry – Opacity Hull

- Visual hull augmented with view-dependent opacity



Geometry – Opacity Hull

- Visual hull augmented with view-dependent opacity
- Store the opacity of each observation at each point on the visual hull.



Opacity Hull

Outline

- Overview
- Previous Works
- Geometry
- Reflectance
- Rendering
- Results

Light Transport Model

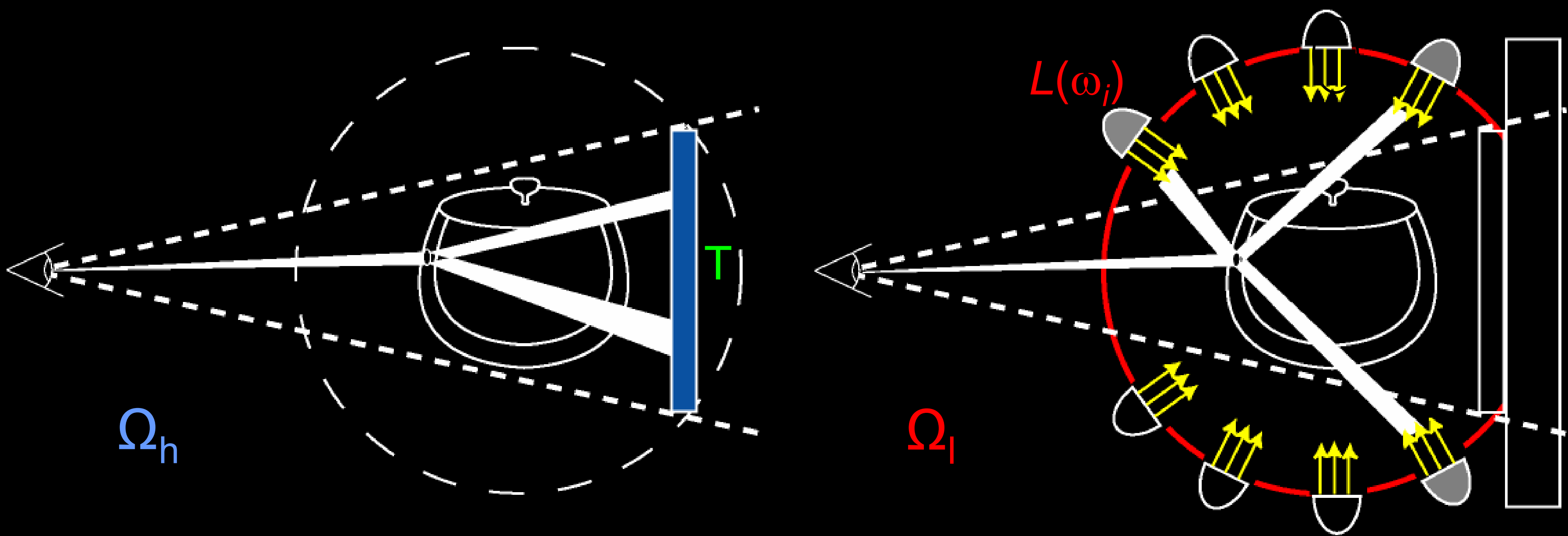
- Assume illumination originates from infinity
- The light arriving at a camera pixel can be described as:

$$C(x,y) = \int_{\Omega} W(\omega) E(\omega) d\omega$$

C(x,y)	- the pixel value
E	- the environment
W	- the *reflectance field*

Reflectance Fields

- Only capture upper hemisphere
- We separate the hemisphere into high resolution Ω_h and low resolution Ω_l.

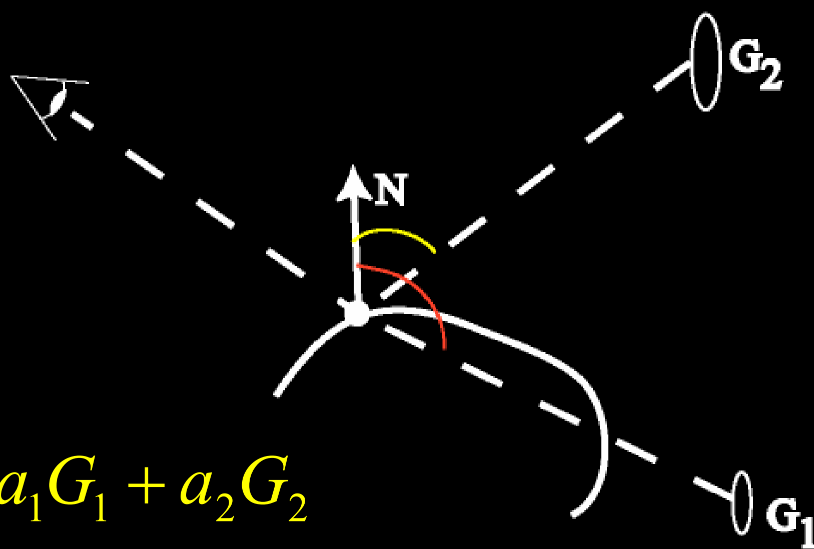


- Now $C(x,y) = \int_{\Omega_h} W_h(\xi) T(\xi) d\xi + \int_{\Omega_l} W_l(\omega_i) L(\omega_i) d\omega$

High-Resolution Reflectance Field (Ω_h)

$$C(x,y) = \int_{\Omega_h} W_h(\xi) T(\xi) d\xi + \int_{\Omega_l} W_l(\omega_i) L(\omega_i) d\omega$$

- Use techniques of environment matting [Chuang et al 00]
- Approximate W_h by a sum of two Gaussians:
  - Refractive G_1.
  - Reflective G_2.



$$W_h(\xi) = a_1 G_1 + a_2 G_2$$

Low-Resolution Reflectance Field (Ω_l)

- W_l sampled by taking a picture with each light turned on at a time [Debevec et al 00]

$$\int_{\Omega_l} W_l(\omega_i) L(\omega_i) d\omega \approx \sum_{i=1}^{n} W_i L_i$$

for n lights

Acquisition

- For each viewpoint (6 cameras x 72 positions)
  - Alpha mattes
    - Use multiple backgrounds [Smith and Blinn 96]
  - Reflectance images ← Low resolution
    - Pictures of the object under different lighting (4 lights x 11 positions)
  - Environment mattes ← High resolution
    - Use similar techniques as [Chuang et al. 2000]

Outline

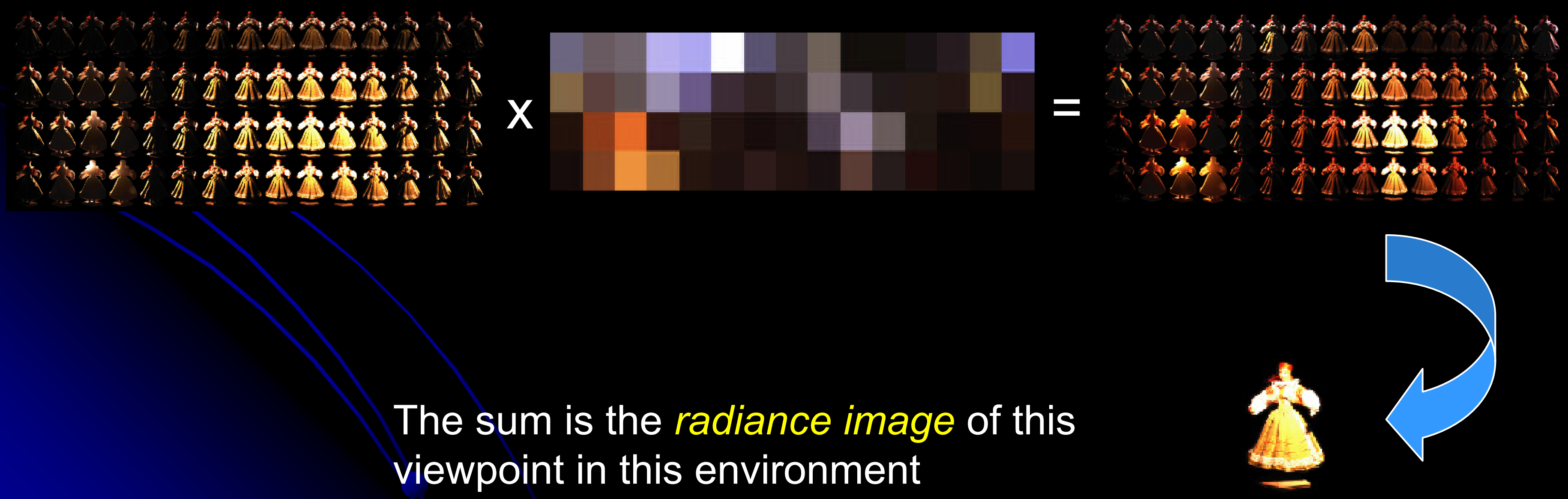
- Overview
- Previous Works
- Geometry
- Reflectance
- Rendering
- Results

Rendering

- Input: Opacity hull, reflectance data, new environment
- Create *radiance* images from environment and low-resolution reflectance field
- Reparameterize environment mattes

Rendering – Radiance Image

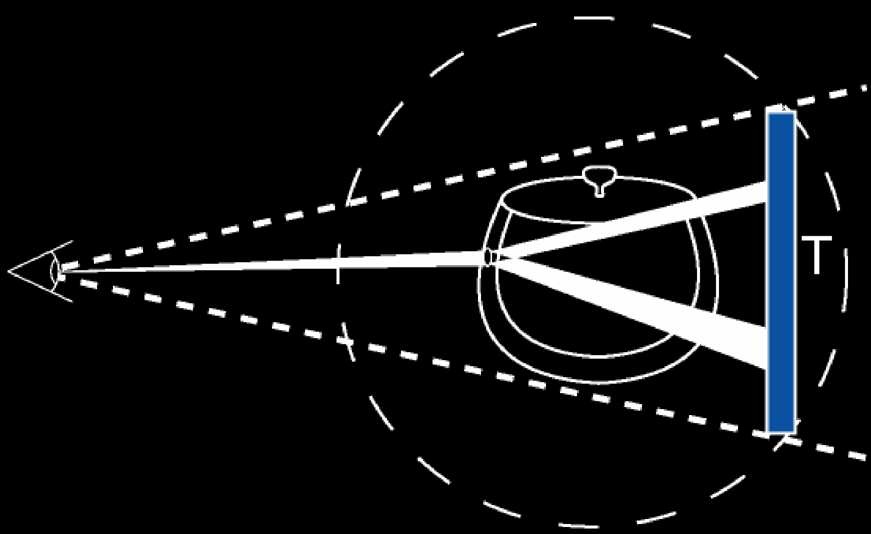
- Downsample environment corresponding to Ω_l
- Compute *radiance image* for each viewpoint

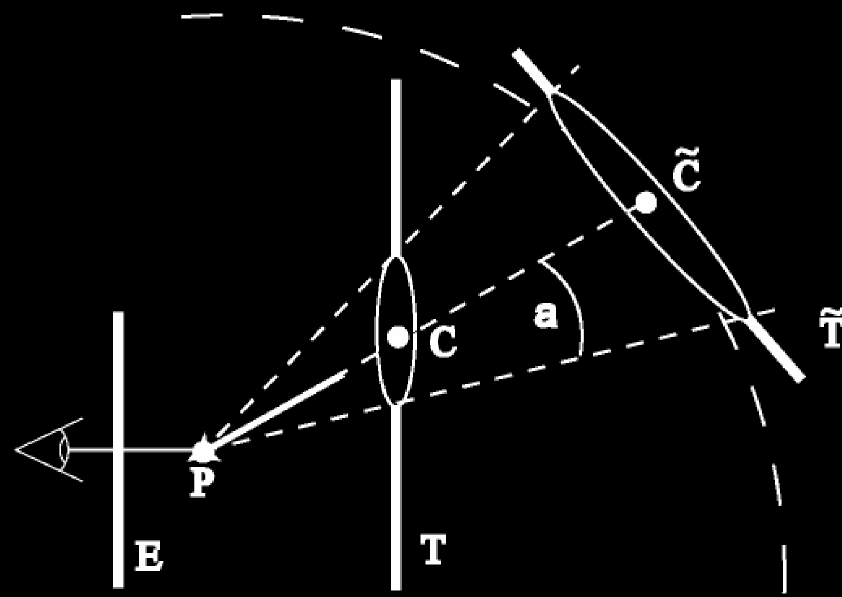


The sum is the *radiance image* of this viewpoint in this environment

Rendering

- Project environment mattes onto the new environment
  - Environment mattes parameterized on plane T.
  - Need to project the Gaussians to the new environment map





Rendering

- From new viewpoint, for each surface element, find 4 nearest viewpoints
- Interpolate using unstructured lumigraph [Buehler et al, SIGGRAPH 01]
  - Opacity
  - Contribution from low-res reflectance field (in the form of radiance images)
  - Contribution from high-res reflectance field

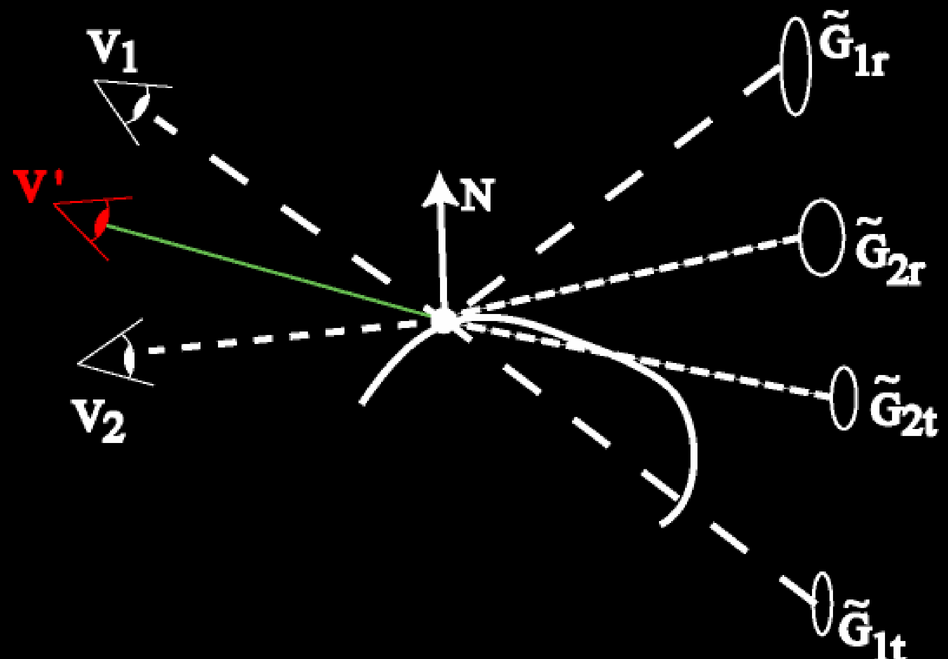
Rendering

- For low-res reflectance field, we interpolate the RGB from the radiance images

$$p' = w_1 p_1 + w_2 p_2 + w_3 p_3 + w_4 p_4$$

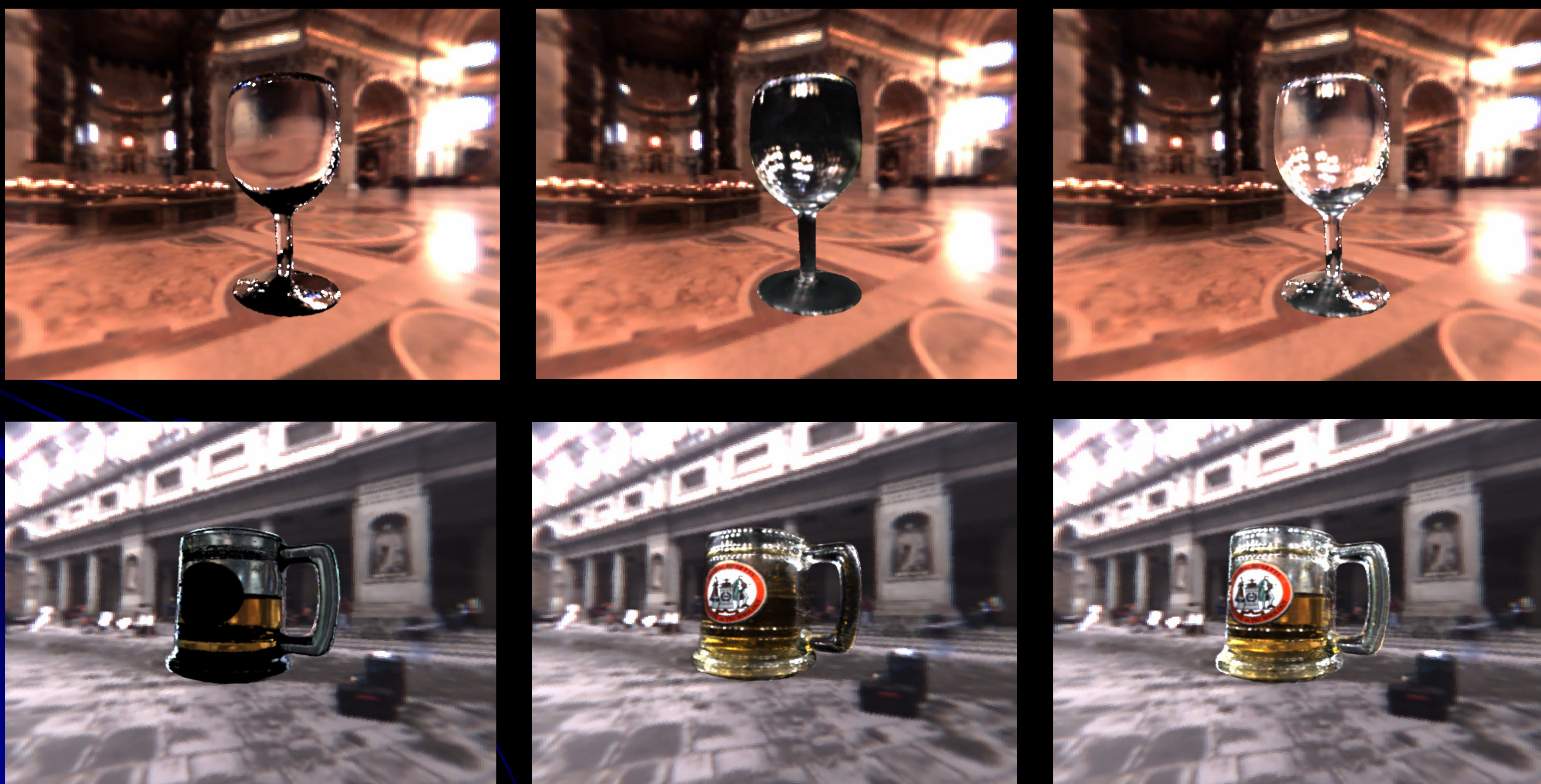
Rendering

- For high-resolution reflectance field:
  - Interpolate *direction* and other params of reflection/refraction Gaussians
  - Convolve with the environment



Results

High-resolution Ω_h Low-resolution Ω_l Combined



Results

Results

Results

Results

Results

Results

Results

- Performance for 72x6 = 432 viewpoints
- 337,824 images taken in total !!
  - Acquisition (47 hours)
    - Alpha mattes – 1 hour
    - Environment mattes – 18 hours
    - Reflectance images – 28 hours
  - Processing
    - Opacity hull ~ 30 minutes
    - PCA Compression ~ 20 hours (unoptimized MATLAB code)
  - Rendering ~ 5 minutes per frame
  - Size
    - Opacity hull ~ 30 – 50 MB
    - Environment mattes ~ 0.5 – 2 GB
    - Reflectance images ~ Raw 370 GB Compressed 2 – 4 GB

Future Work

- Use more than 2 Gaussians for the environment mattes.
- Better compression.
- Faster/real-time rendering.

Conclusions

- A fully automatic system that is able to capture and render transparent and refractive objects.
- Separation of surface reflectance fields into high- and low-res areas practical
- New rendering algorithm for view interpolation

Acknowledgements

- Thanks to:
  - Frédo Durand
  - MIT Graphics Group, MERL colleagues
  - NSF Grants CCR-9975859, EIA-9802220