RAVES Rendering January 2005 Update

Sumanta N. Pattanaik
School of Computer Science

Topics

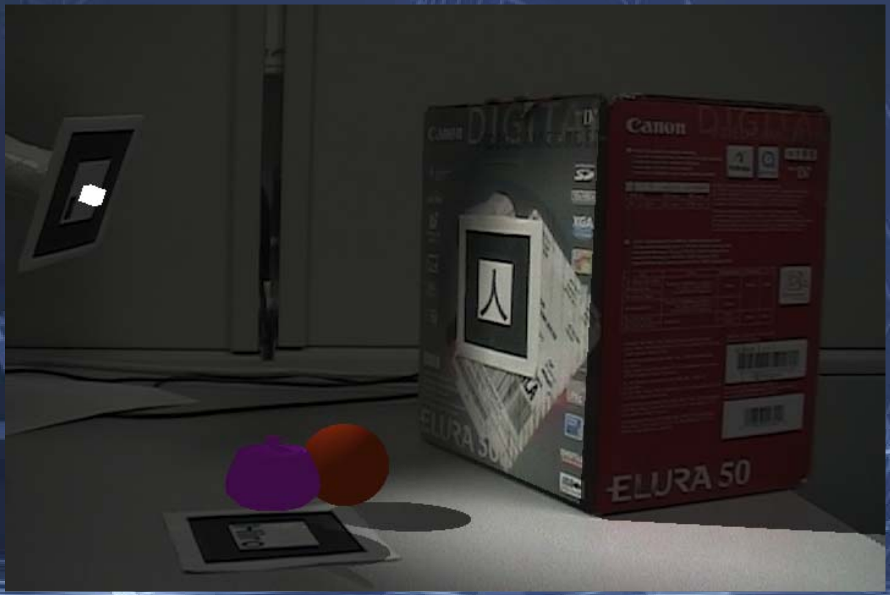
- Rendering for MR
- Real-Time Rendering algorithms
- Future (spring 2005)
- PhD Students
- Publications

Rendering for MR

- Key Issues
  - Lighting and shadow
    - Virtual object illuminated by real light
    - Real object illuminated by virtual light
    - Shadows of virtual on real, and of real on virtual
  - Color Matching
  - Real Time (interactive rate) computation

Importance of Shadows

- Shadows are an important part of lighting simulation:
  - Lights should cast shadows on real objects even though the light sources are virtual.

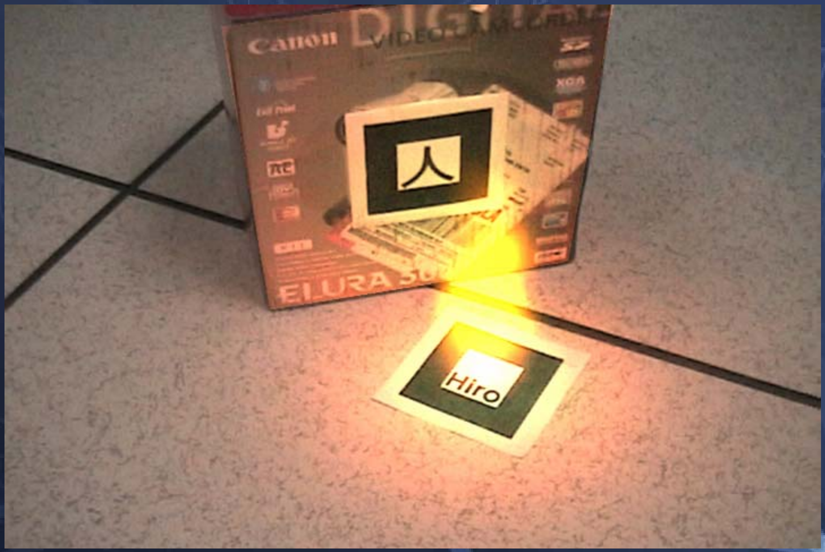


Importance of Shadows

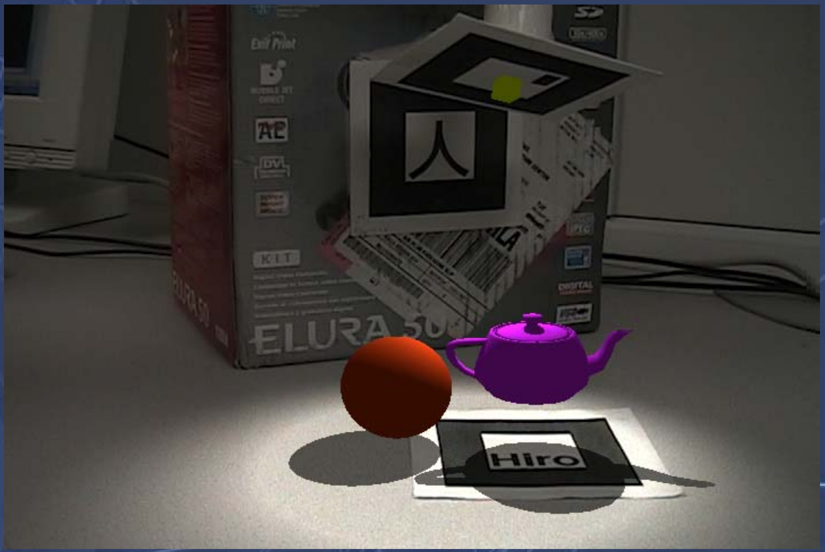
- Helps us understand:
  – Relative virtual-to-real object position and height in space.
  – Shape of a virtual shadow receiver's surface.
  – Position of nearby virtual objects that are occluded or outside the field of view (virtual enemy in hiding).

Our Contribution

- We demonstrate two ways to inject virtual light into a scene in the context of two practical MR applications.



Virtual Fire



Virtual Flashlight

Occlusion Models

• Our algorithms make use of the occlusion models of real objects which are common in MR applications.

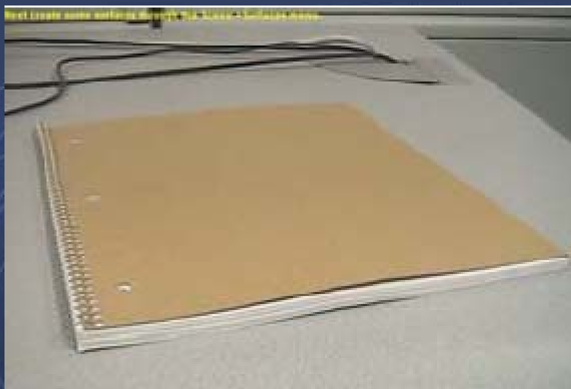
• Occlusion models describe the geometry and dimensions of all interactive real objects in the scene. They can be pre-measured or automatically generated.

• Model position relative to camera must be tracked in some way.

• Usually used for determining which parts of virtual objects are hidden by real objects closer to the camera.

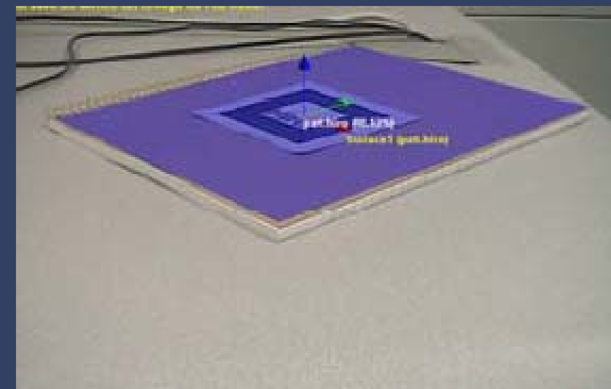
• Having this geometry available lets us adapt many algorithms from computer graphics.

Occlusion Models



No occlusion model

With occlusion model



Tracking the model's position

- Example of the occlusion model for a notebook (approximated by a single polygon).

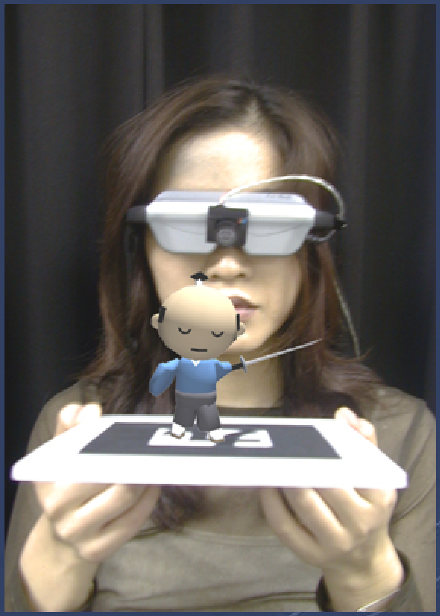
- Tracked by a marker.

Tracking

- Need to know position relative to camera of every interactive real object.
- Any object that is allowed to move independently of others must also be tracked independently.
- Several methods of tracking:
  - Sensor-based (InterSense, Polhemus, GPS)
  - Image-based (ARToolkit)

Tracking

- ***ARToolkit*** is an image-based tracker which derives camera position relative to a particular marker (and vice versa) based on its location and tilt in the video frame.
- Developed by University of Washington’s Human Interface Technology Lab.
- We chose ***ARToolkit*** for our tests for several reasons:
  - Light-weight
  - Free
  - Relatively easy to set-up
  - No equipment to drag around

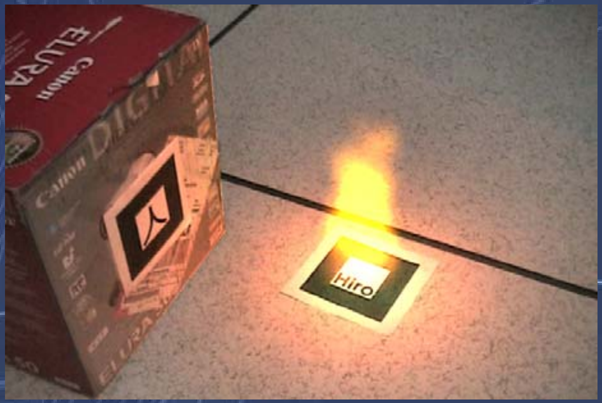


Virtual Fire

- Our virtual fire application simulates a real environment light by computer-generated fire.
- The process works by calculating how the intensity of each pixel covered by an occlusion model is increased by the virtual flames.

Unmodified frame

Virtual illumination only



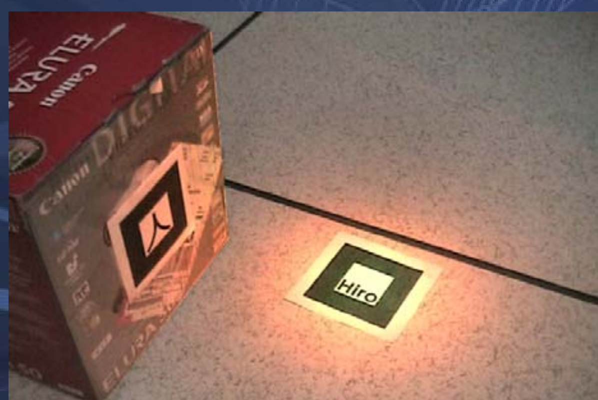
Illumination + flames

Virtual Fire

- Thanks to occlusion models, this calculation can be done with many known computer graphics lighting algorithms and can be accelerated by graphics cards. The calculation can be simple or complex depending on the requirements of the application.

- The original intensity of the pixel is scaled up based on the result of the computation. Finally, virtual flames are drawn onto the image.

Unmodified frame

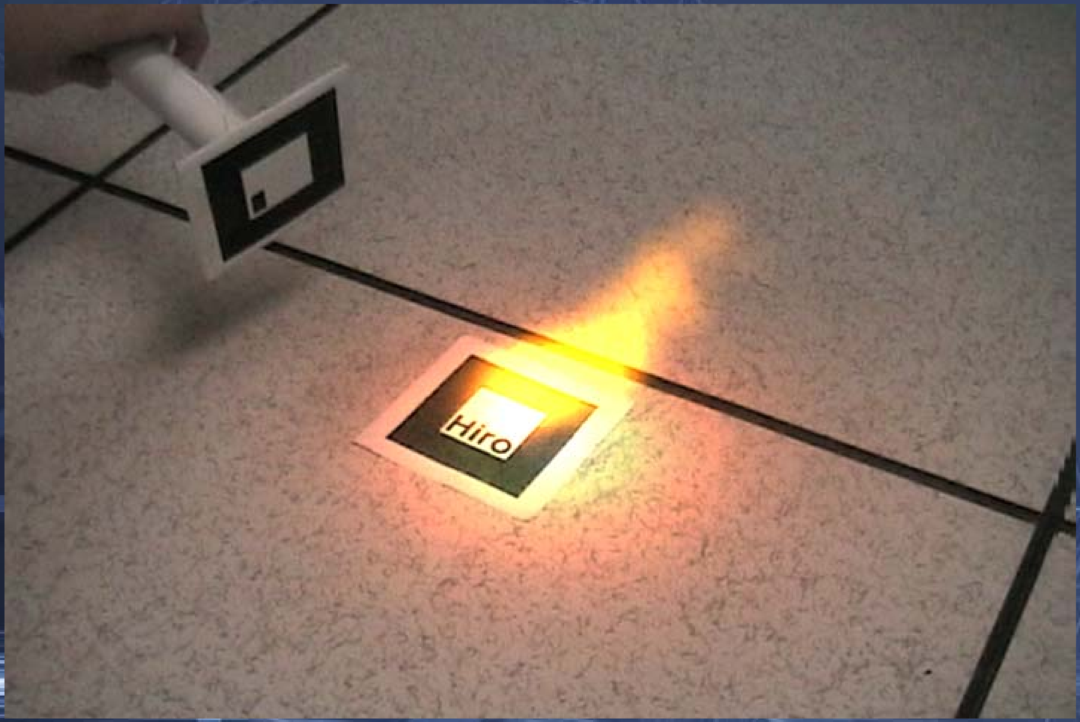


Virtual illumination only

Illumination + flames

Virtual Wind

- A separate marker tracks the position of a non-existing fan. Lighting on ground shifts with flames' position.



Virtual Flashlight

- Artificially make a room darker, and restore it to its original intensity with a virtual flashlight.

- Done in multiple steps.

Unmodified video frame

Final processed frame

Virtual Flashlight

- **Step #1**: Scale down intensity of pixels that are ***not*** covered by tracked occlusion models by some constant factor.
- *Untracked pixels assumed to be unlit by virtual light.*

Partially darkened frame

Final processed frame

Virtual Flashlight

- **Step #2**: Calculate shadows from virtual objects falling on real objects.
- *Using a version of the shadow volume algorithm from computer graphics adapted for MR.*

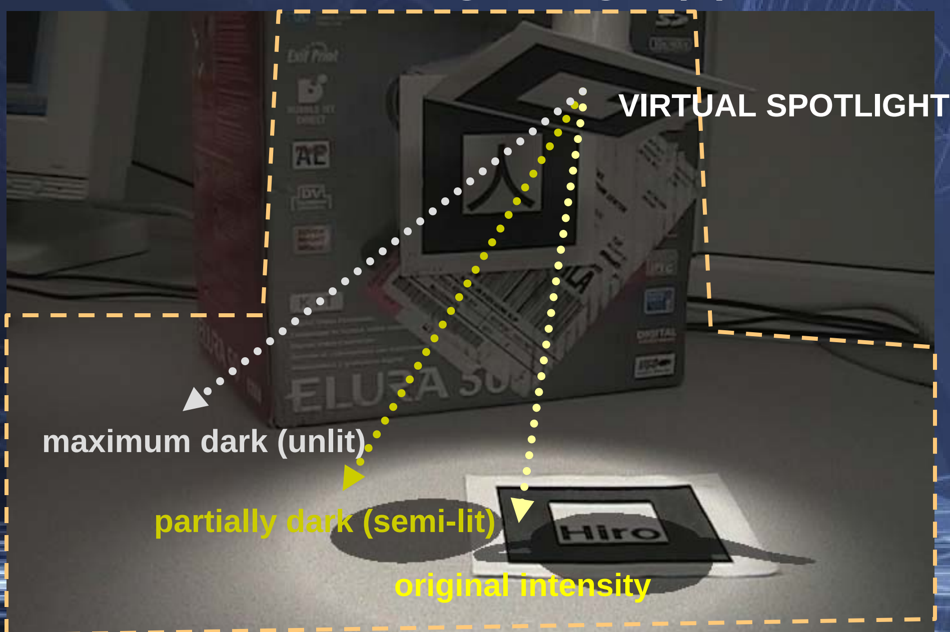
Virtual shadows on real objects

Final processed frame

Virtual Flashlight

- **Step #3**: Artificially darken occlusion model -covered pixels, and simulate virtual light by darkening virtually "lit" pixels less.
- *Modified intensity is determined by the spotlight algorithm from computer graphics.*

Full virtual lighting applied

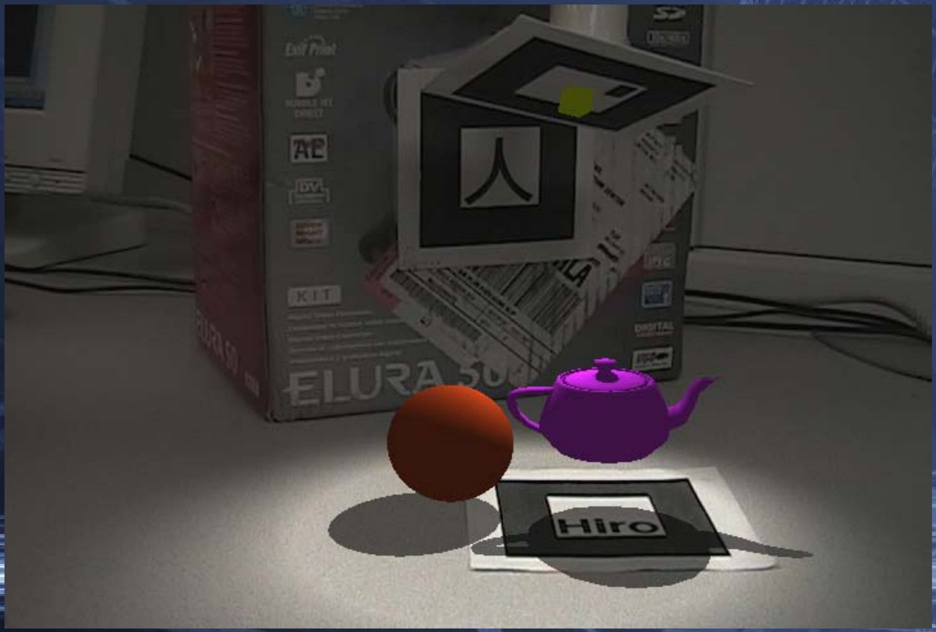


Final processed frame

Virtual Flashlight

- **Step #4**: Composite virtual objects and compute shadows from real and other virtual objects falling on them.
- *Use same basic algorithm as in Step #2.*

Final processed frame

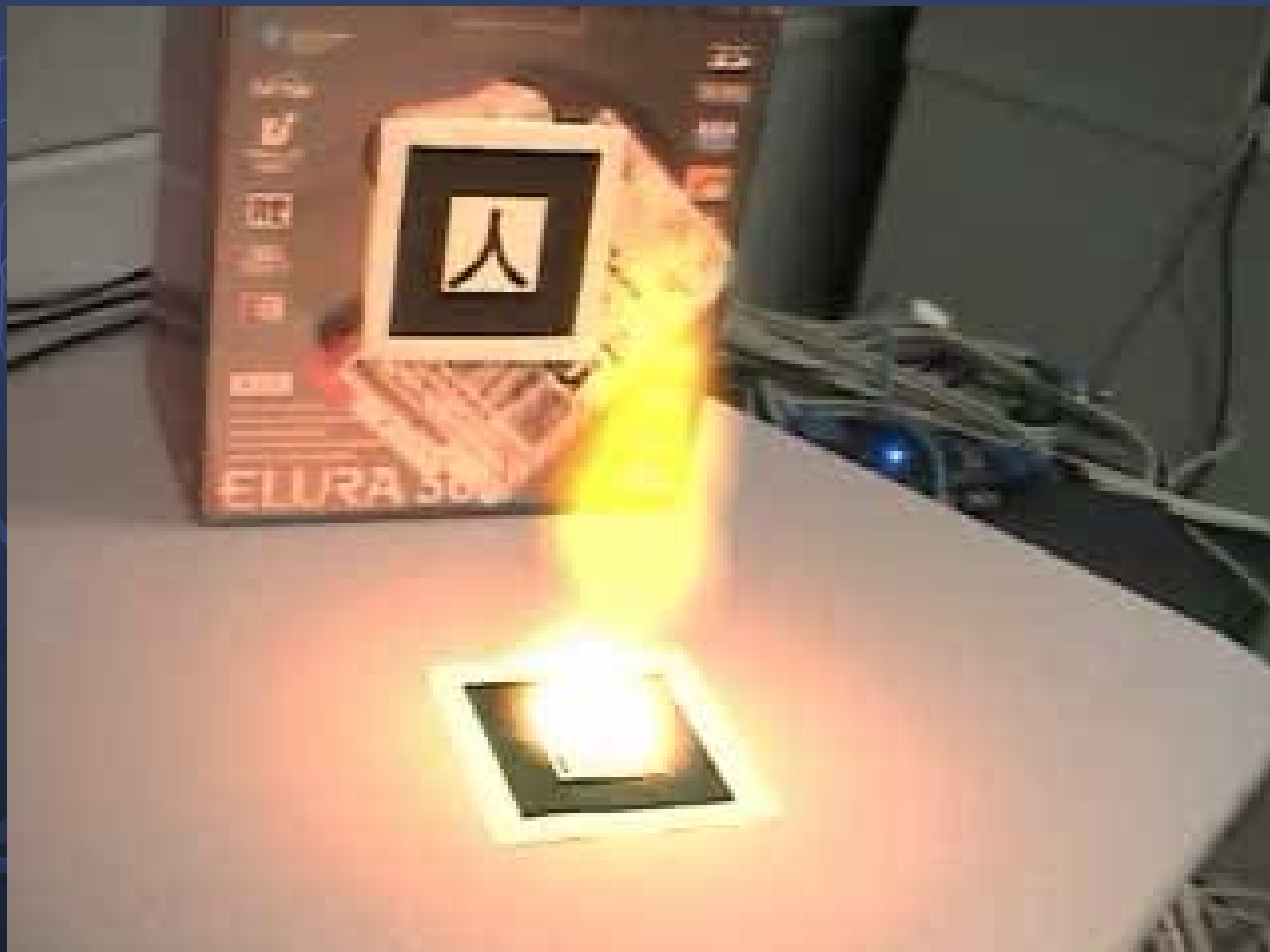


Final processed frame

University of Central Florida

R.A.V.E.S

Video



(click movie to begin)

Real Light

Whole environment is the light source

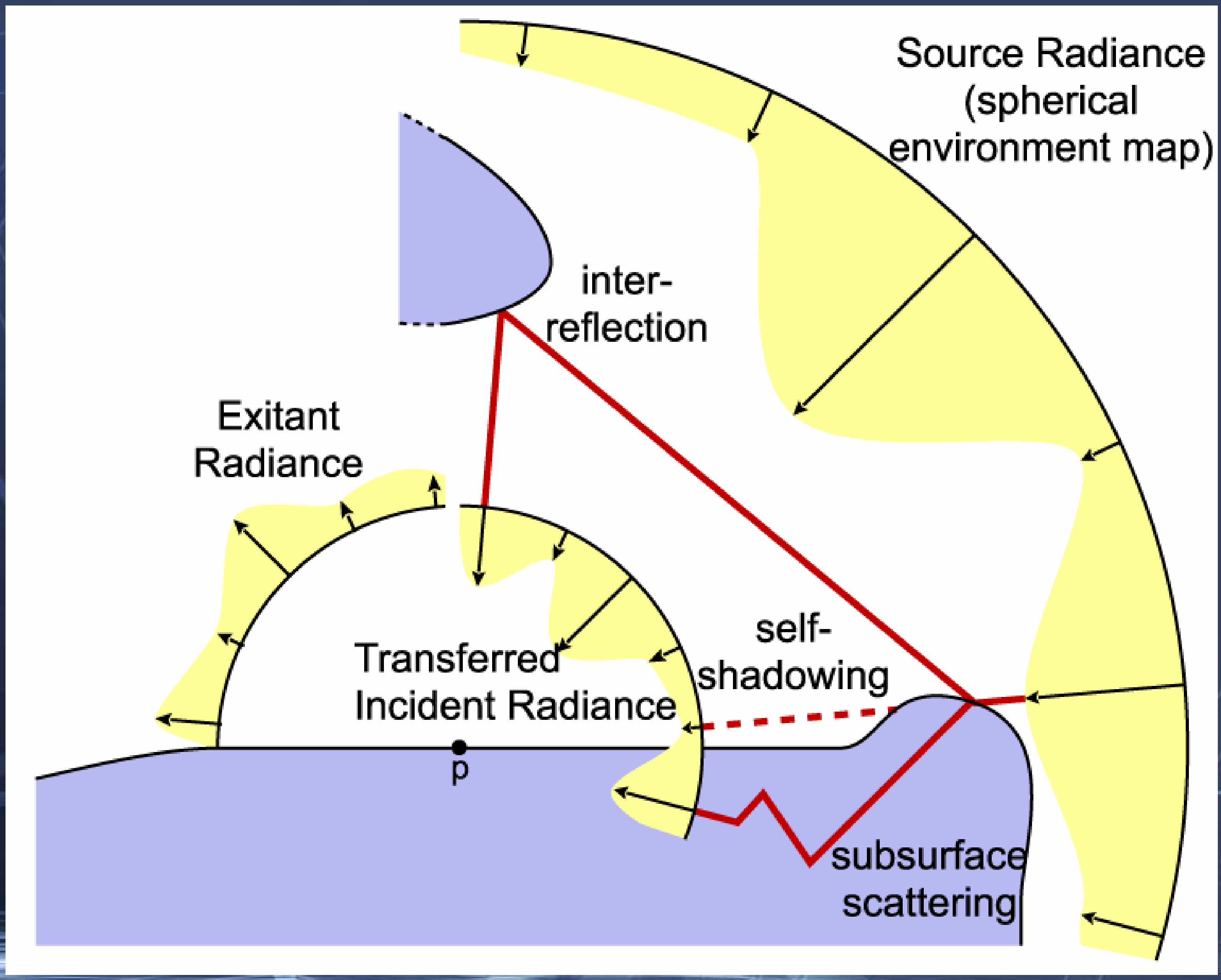
- Issues:
  - Capture
  - Render Virtual object with the real light
    - Lighting
    - Shadow

(Ph.d. Student: Ruifeng)

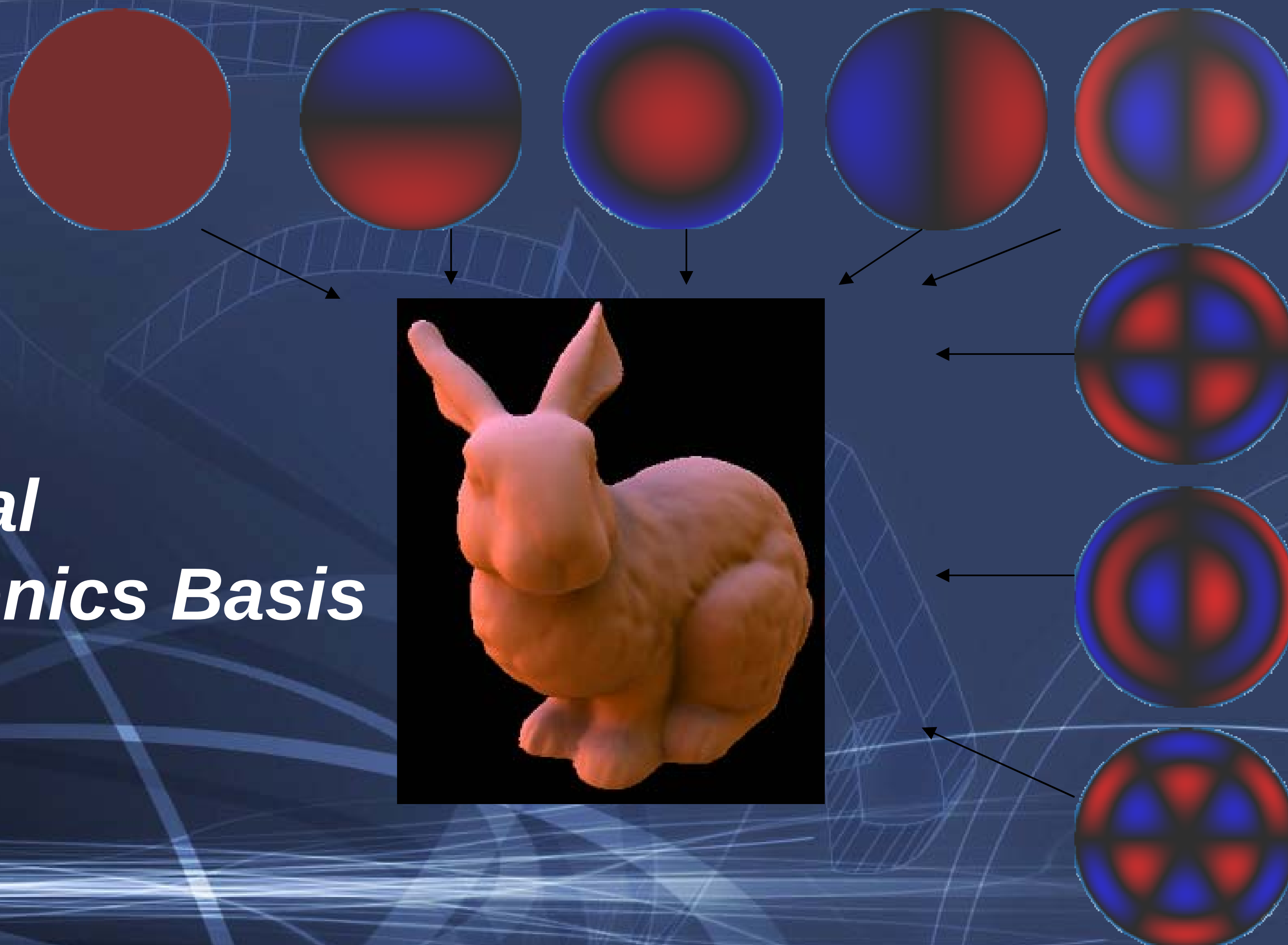
Real Light Capture

- Instrument the environment
  - Ladybug
  - Upto 15 fps per second capture capability (dynamic light capture possible).

Rendering Using Environment Light



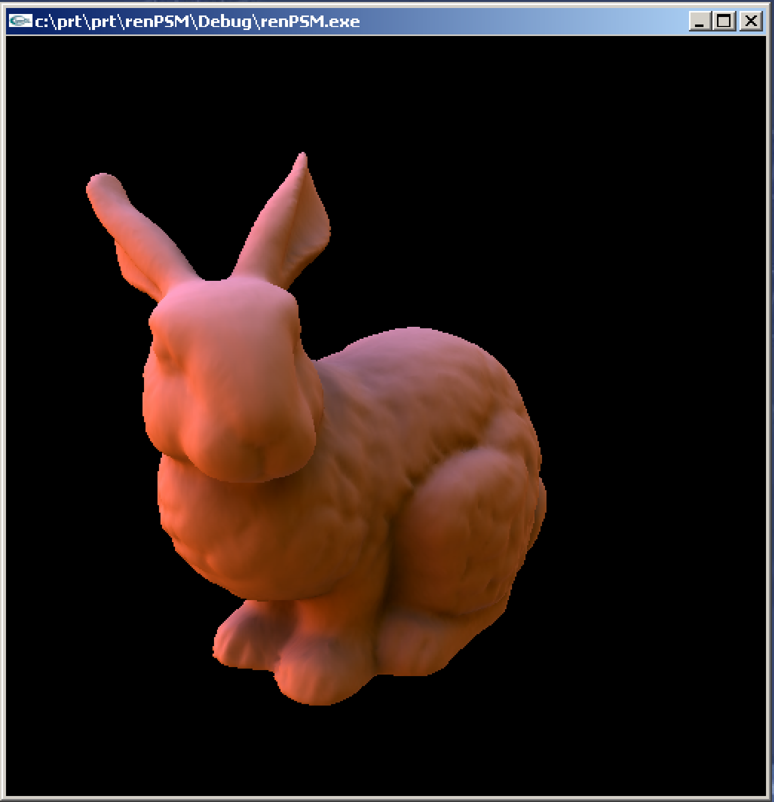
Pre-computation using Basis Light Functions

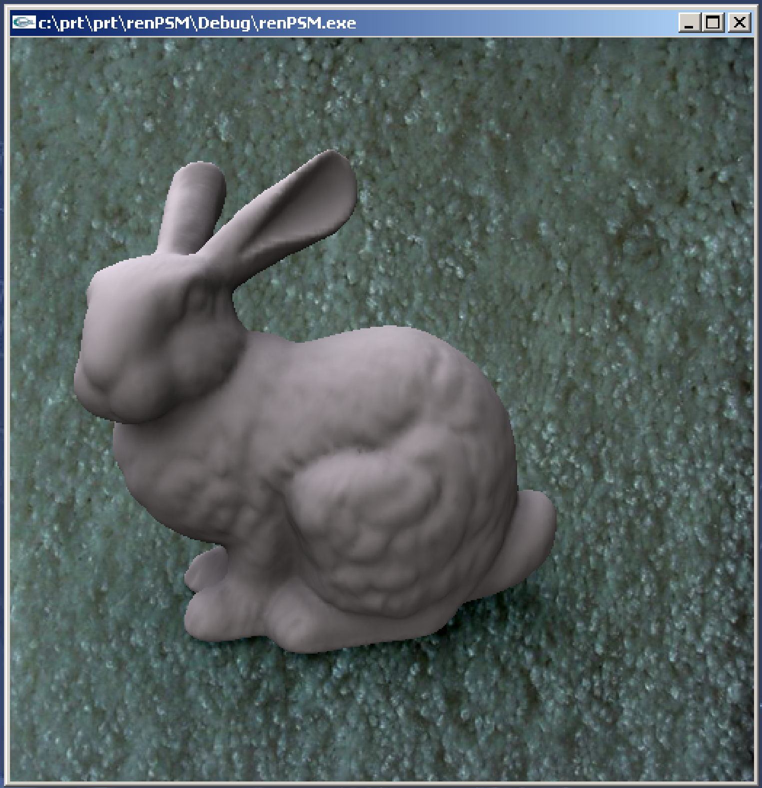


Spherical Harmonics Basis

Lighting & Shadows

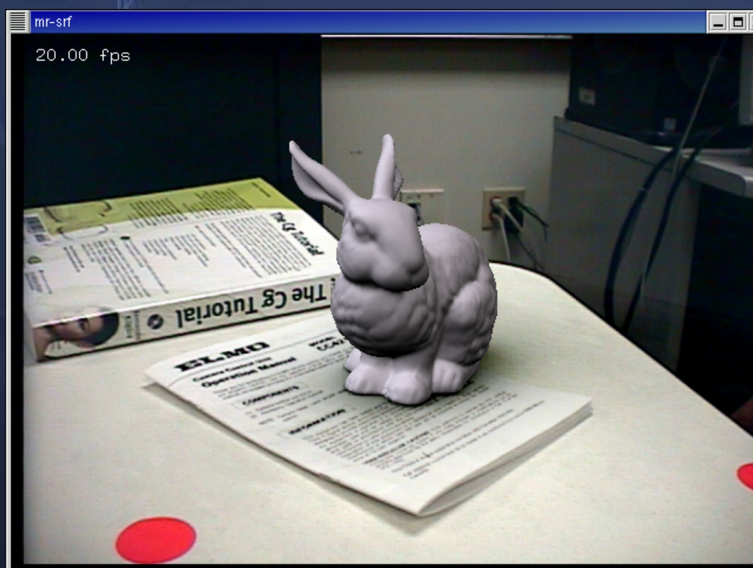
- Self shadows by Precomputed Radiance



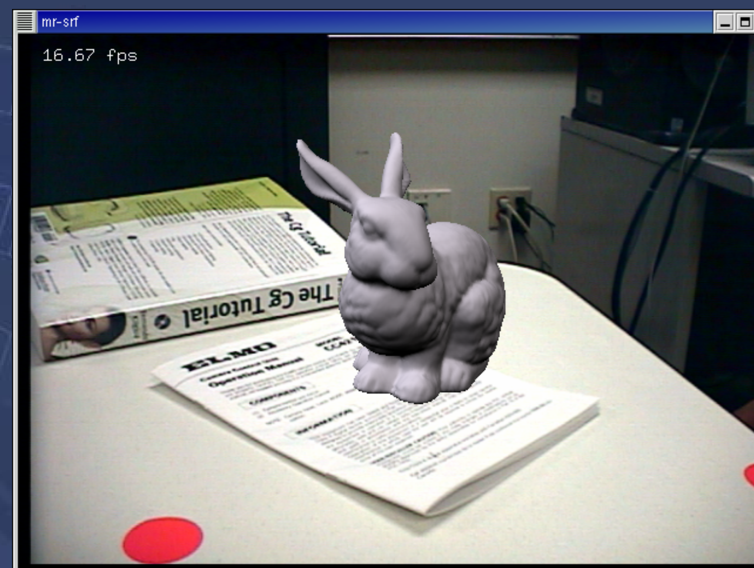


Lighting & Shadows

- Shadows from virtual objects to real world



With shadow

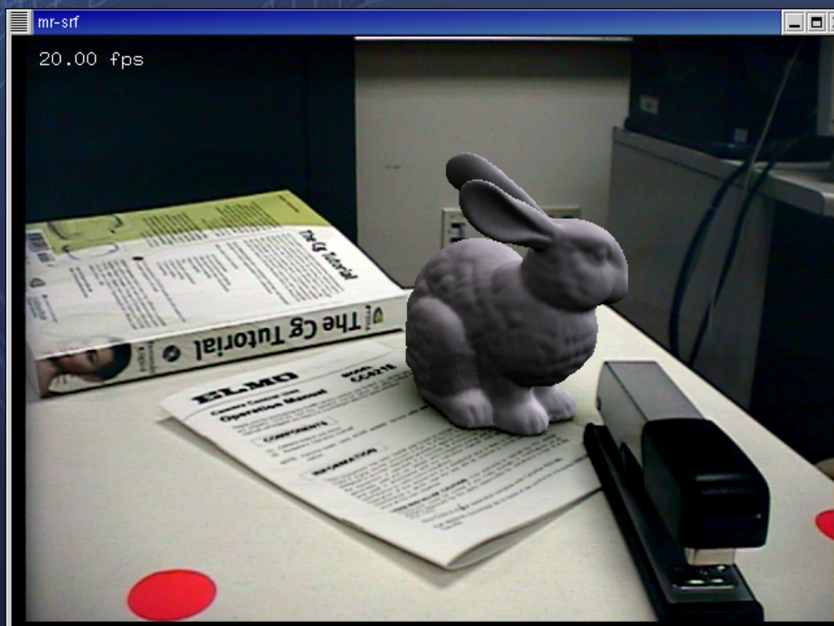


without shadow

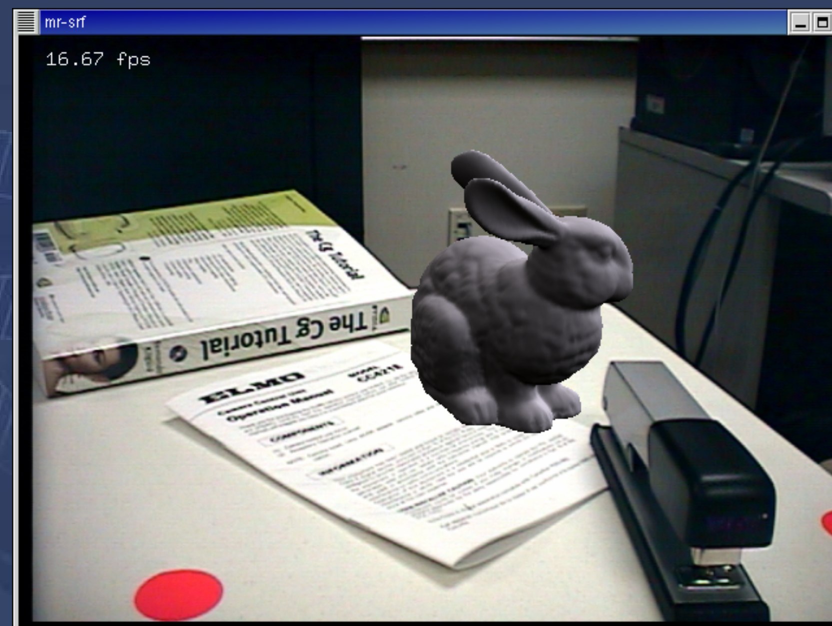
University of Central Florida

R.A.V.E.S

Lighting & Shadows



with shadow



without shadow

- Strengths
  - Soft shadow
  - Real area light sources
  - Rendered in real time (7-35 fps for 70K triangles)

Dynamic Object Rendering

Problem

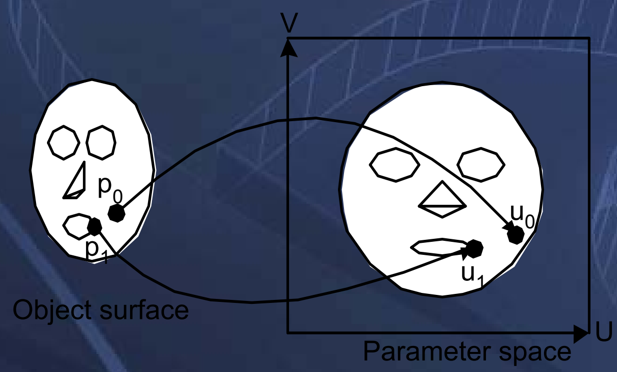
- Object may deform
- Pre-compute light map for each frame/pose generates huge amount of data. Sheer data size hinders real-time processing.

Idea

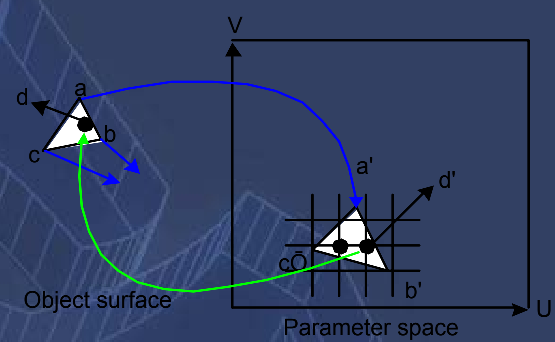
- Record on unfolded 2D map
- Compress 2D map

Dynamic Object Rendering

- Record PRT data on unfolded parameter space



Parameterize 3D surface



Mapping of non-vertex points

Dynamic Object Rendering

- results

only lighting with lighting and texture PRT Map

Object	OFW (2811 vertices; 2197 triangles)
Action	2 sec. of walk (100 frames)
Pre-computation time of GI computation	1.5 hours for 100 frames
Raw video data	>100 MB (RGBE format)
Compressed HDR video size	3.5M
Rendering speed	>10 frames/sec

Color Processing

Problem:

Color of virtual objects do not match color of real world.

Solution:

Approach 1: (expensive, accurate result)
Model virtual surface reflectance accurately,
Capture the real-world lighting accurately,
Render the virtual object using the captured virtual light.

Approach 2: (inexpensive, plausible result)
Change the color statistics of the virtual object before compositing to match the statistics of the real scene

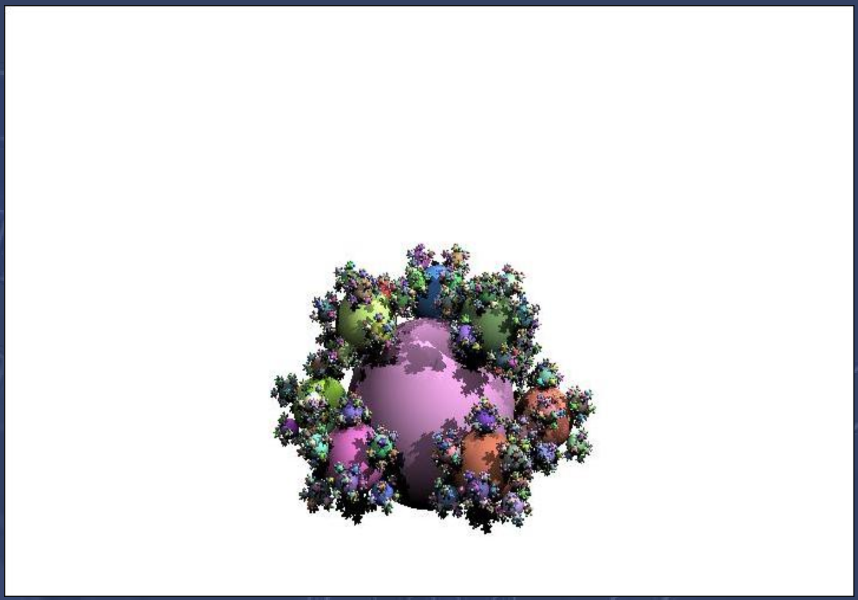
Algorithm

- Covert pixel RGB colors to Opponent color space (Luminance, Chrominance)
- Compute statistics: Mean and Standard Deviation
- Shift and scale pixels of virtual frame to partially match the statistics of Real Frame.
- Covert back opponent colors to RGB

Low Level Optimizations

- Unroll loops by hand
- Compute statistics on a subset of all pixels

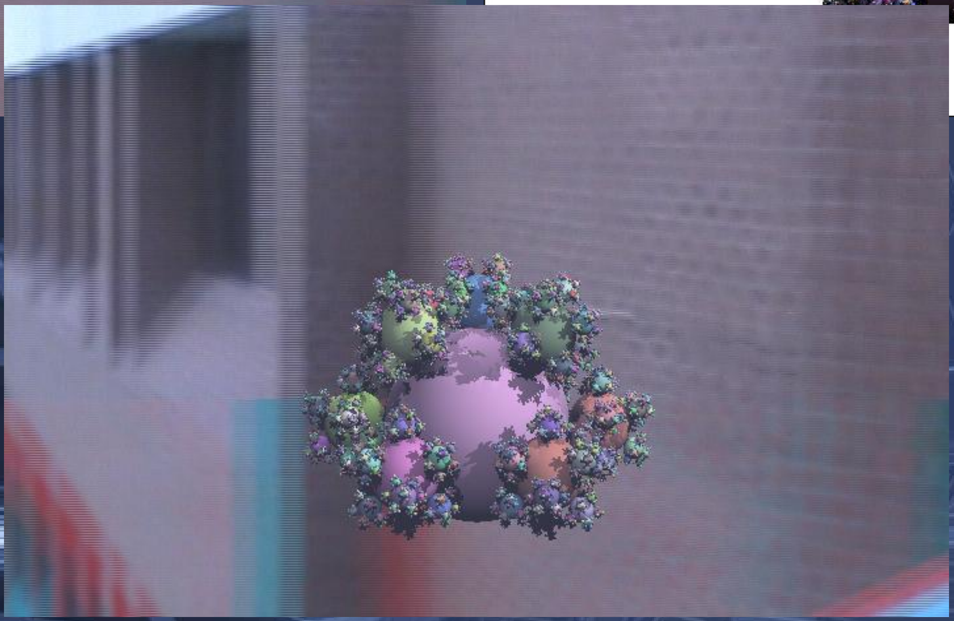
First Test Case



Without Color Adjustment

With Adjustment

Partial Adjustment



Examples

Examples

Video – No Adjustment

University of Central Florida

R.A.V.E.S

Video - Corrected

Conclusions

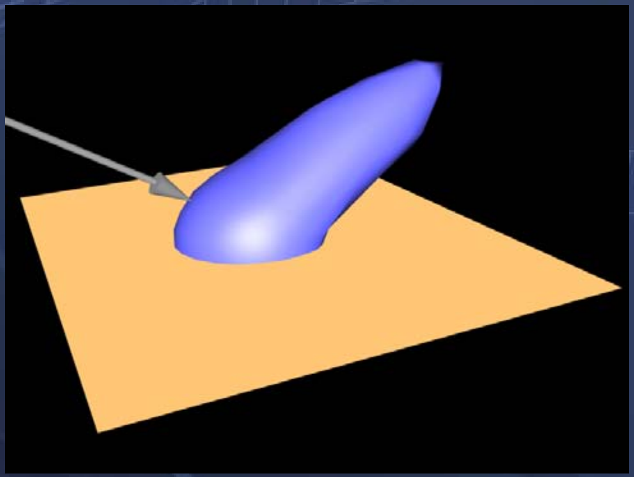
- Simple algorithm, low-level optimizations
- Our video examples run at 22 fps on a 2.53 GHz Pentium – software only

Real-Time Rendering

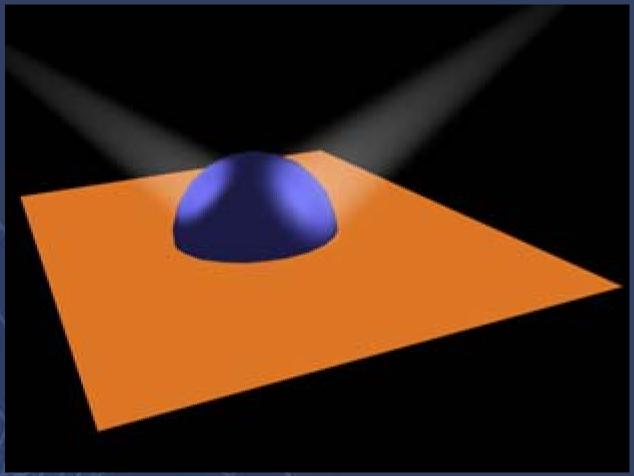
- Key Issues
  - New Algorithm
  - New Data Representation

A Novel Hemispherical Basis for Accurate and Efficient Rendering

Problem Statement



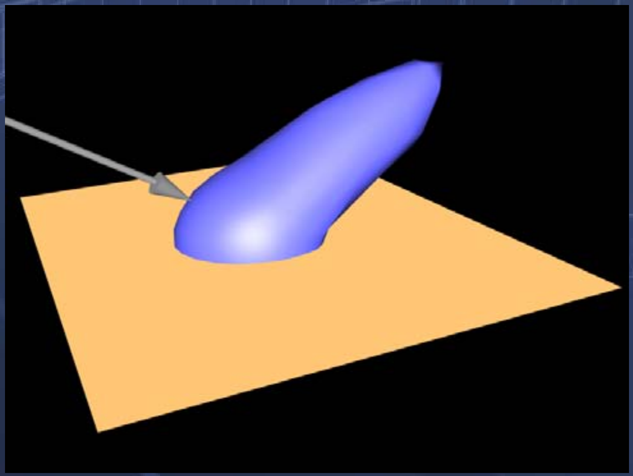
Surface Reflection



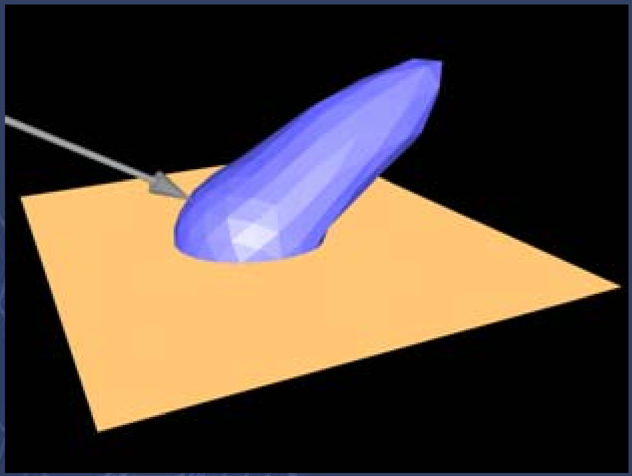
Incoming/Outgoing Light

$$F(\theta, \varphi) \approx \text{Sample set}$$

Problem Statement



$\approx$



Original Function

Piecewise linear approximation

Need a more compact and smoothed representation

Better fitting

Fast computation of integrals

Contribution

- New set of basis functions
  - Formula similar to Spherical Harmonics
  - Designed for representing hemispherical functions
- Applications in lighting simulation

Basis Functions

$$f(x) = f_i \bullet b_i(x)$$

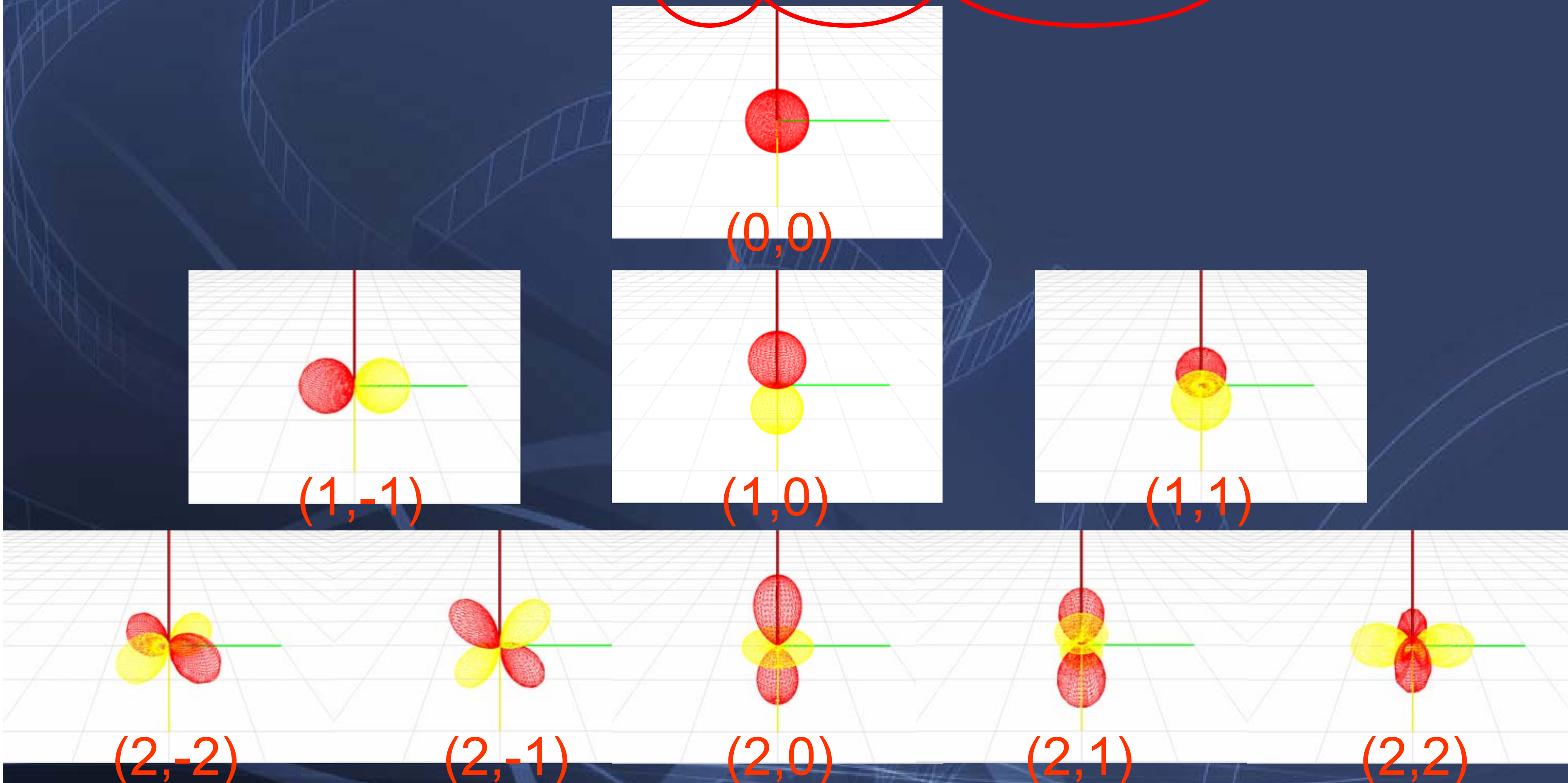
$$f_i = \int f(x) b_i(x) dx$$

$$g(x) = g_i \bullet b_i(x)$$

$$\boxed{\int f(x) g(x) dx = f_i \bullet g_i}$$

Spherical Harmonics

$$Y_l^m(\theta,\varphi) = K_l^m \, \Phi_l^m(\varphi) \, P_l^m(\cos\theta)$$



Spherical Harmonics

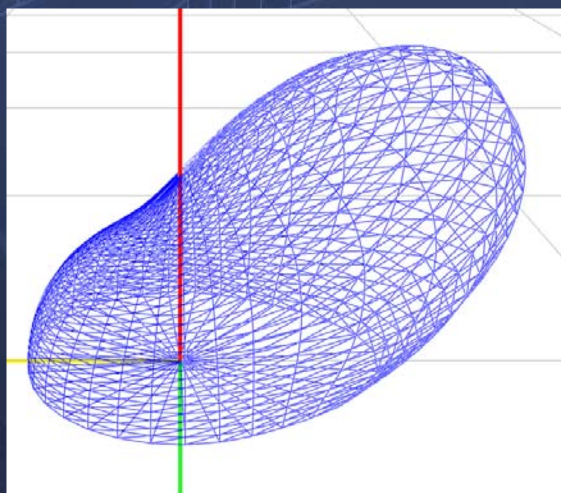
Main Properties

Simple projection and reconstruction

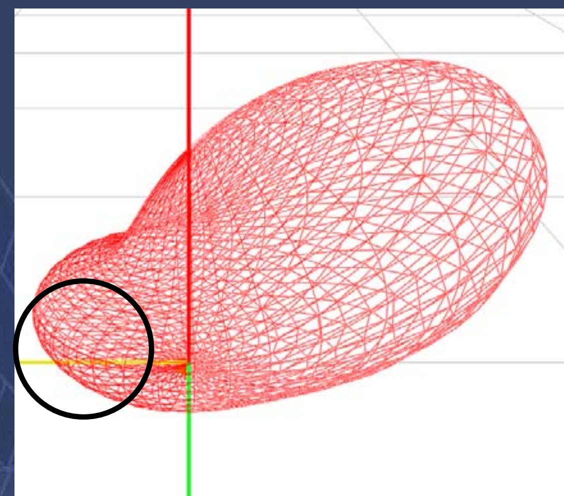
Analytical rotations

SH For Hemispherical Functions

Zero Hemisphere



Original



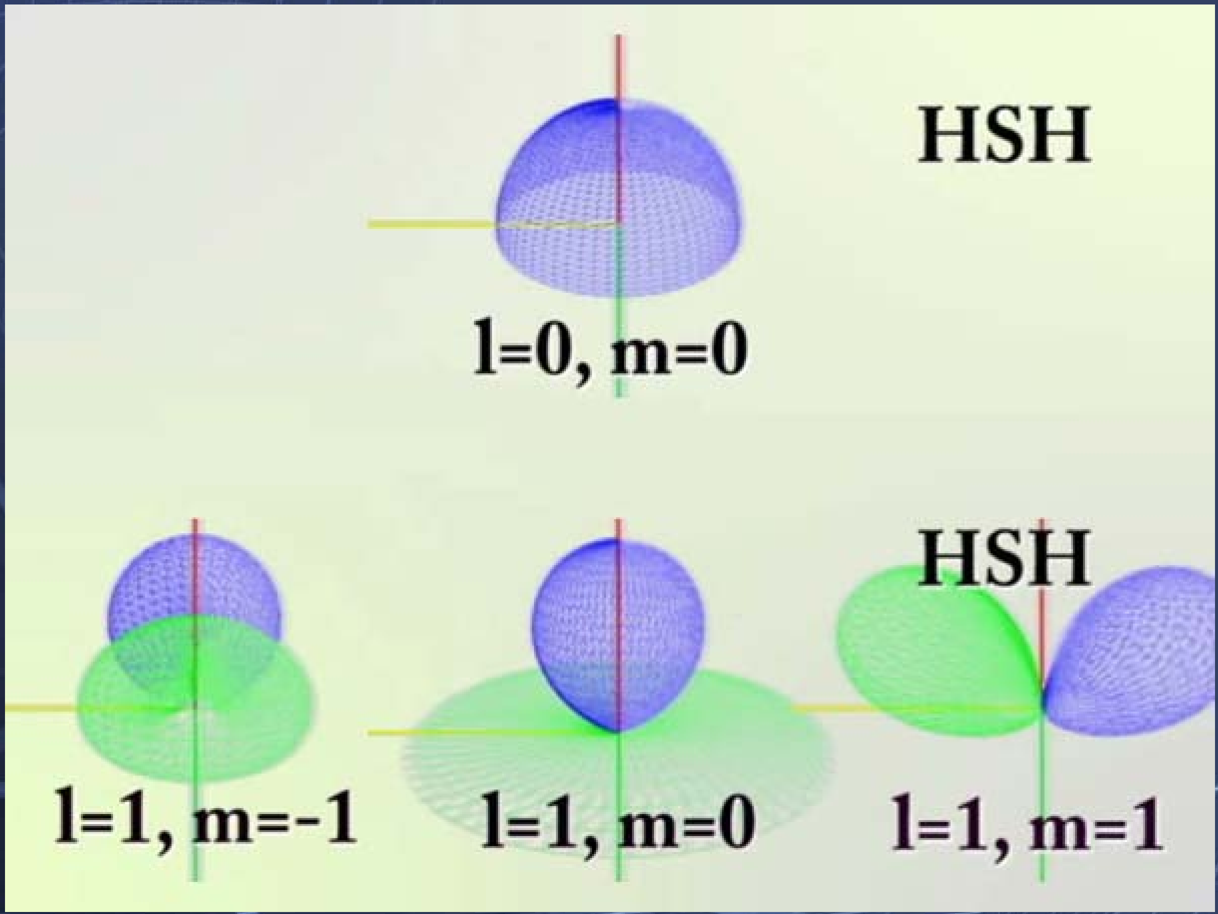
SH

Equator discontinuity

Artifacts

Our Novel Basis

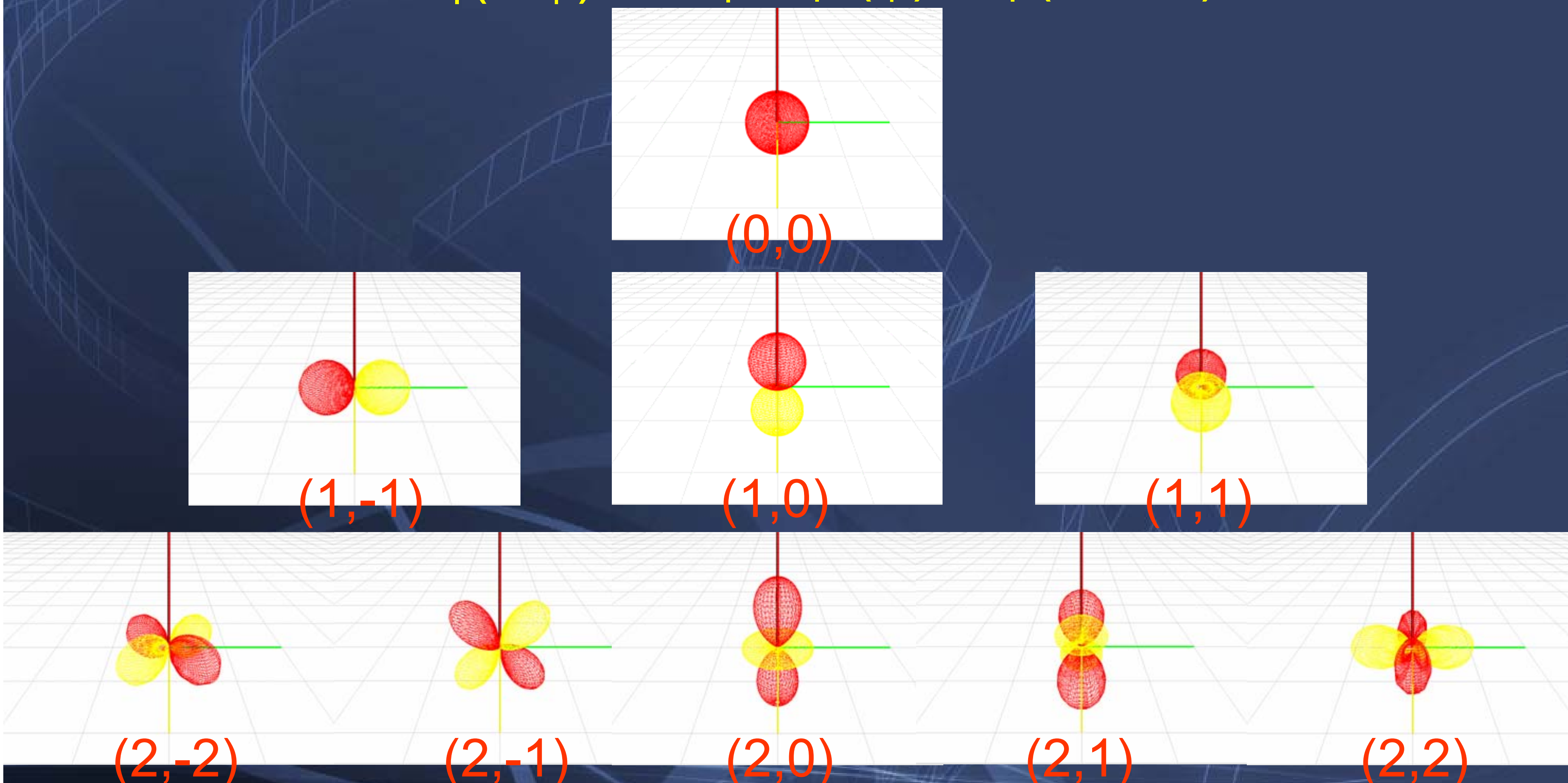
Shifting



Our Novel Basis

Spherical Harmonics

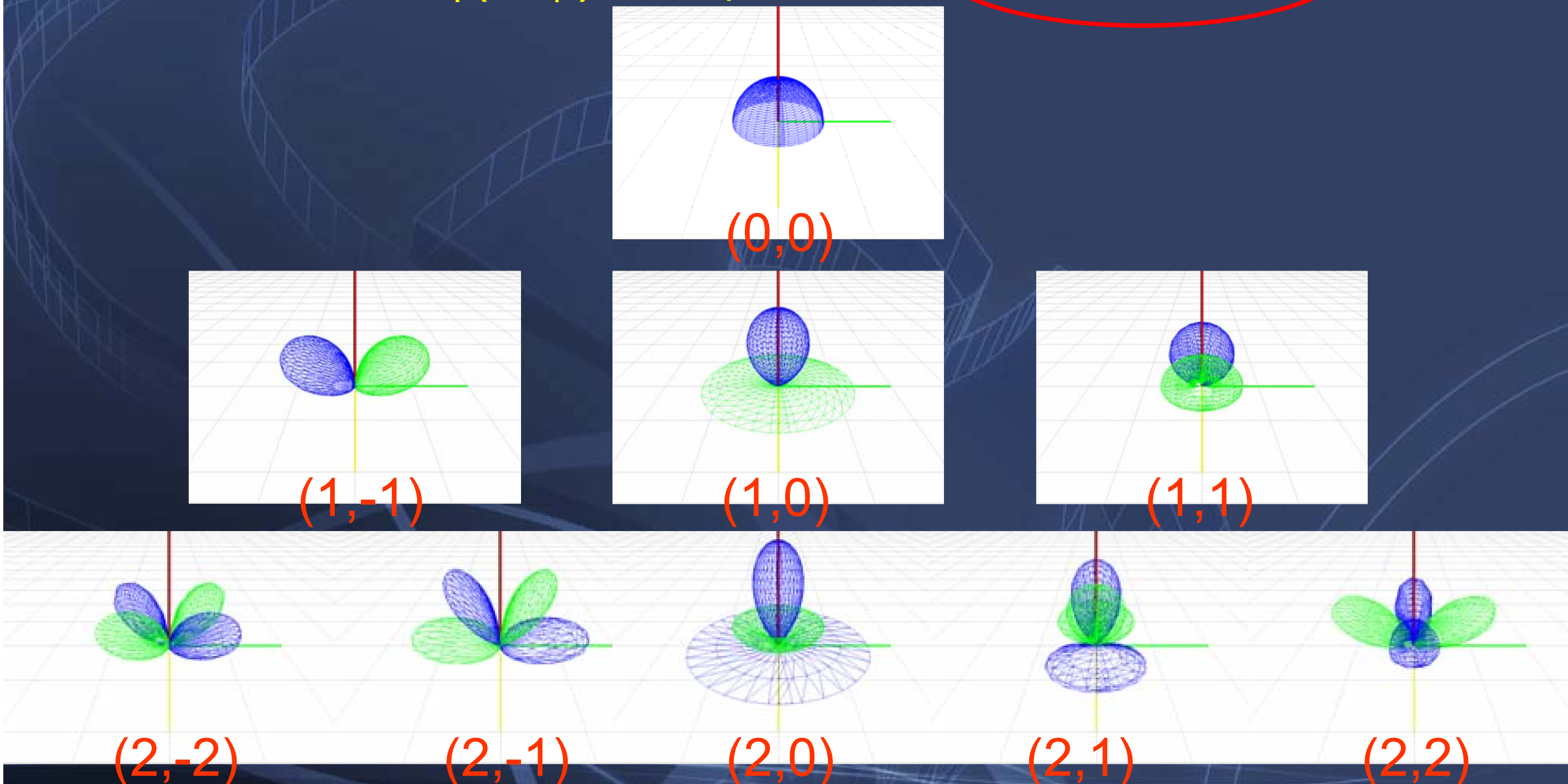
$$Y_l^m(\theta,\varphi) = K_l^m \, \Phi_l^m(\varphi) \, P_l^m(\cos\theta)$$



Our Novel Basis

Hemispherical Harmonics

$$H_l^m(\theta,\varphi) = \tilde{K}_l^m \, \Phi_l^m(\varphi) \, P_l^m(2\cos\theta - 1)$$

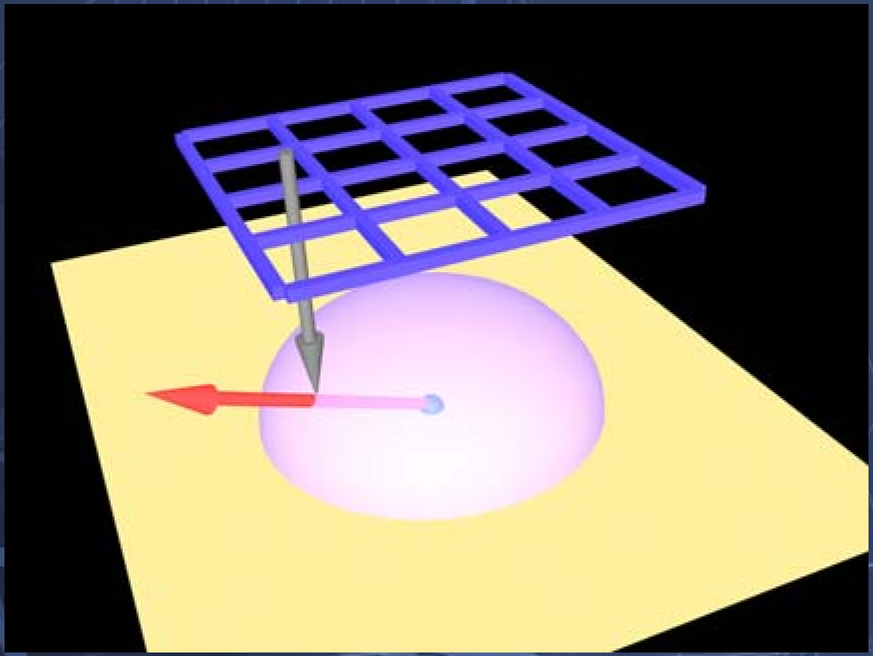


Application: BRDF Representation

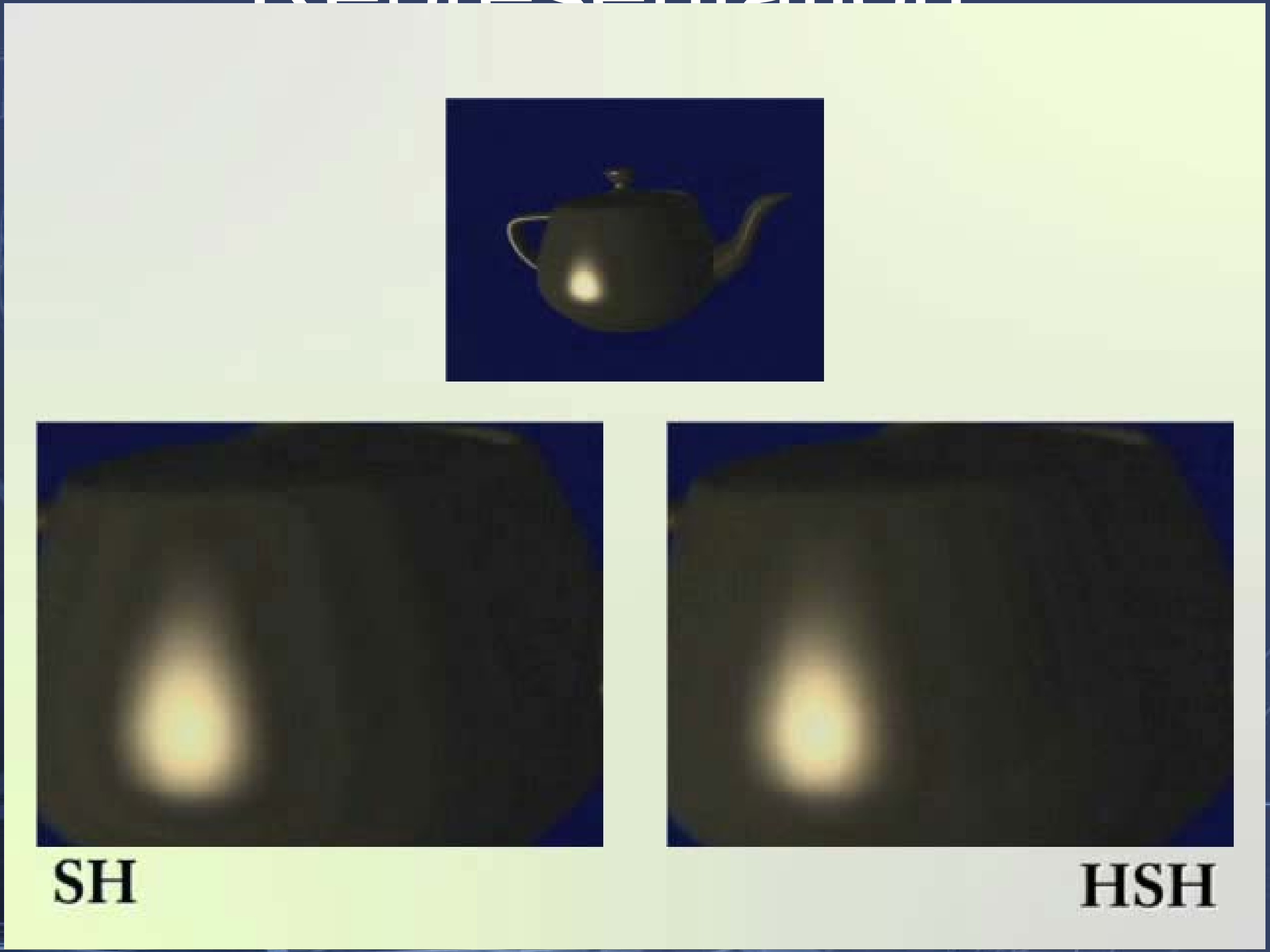
Principle

BRDF = 4D Function

Parabolic Parameterization



Application: BRDF Representation



Application: BRDF Representation

Accuracy

Less Ringing

Higher Frequency

HSSH

Application: Environment Mapping

Performance

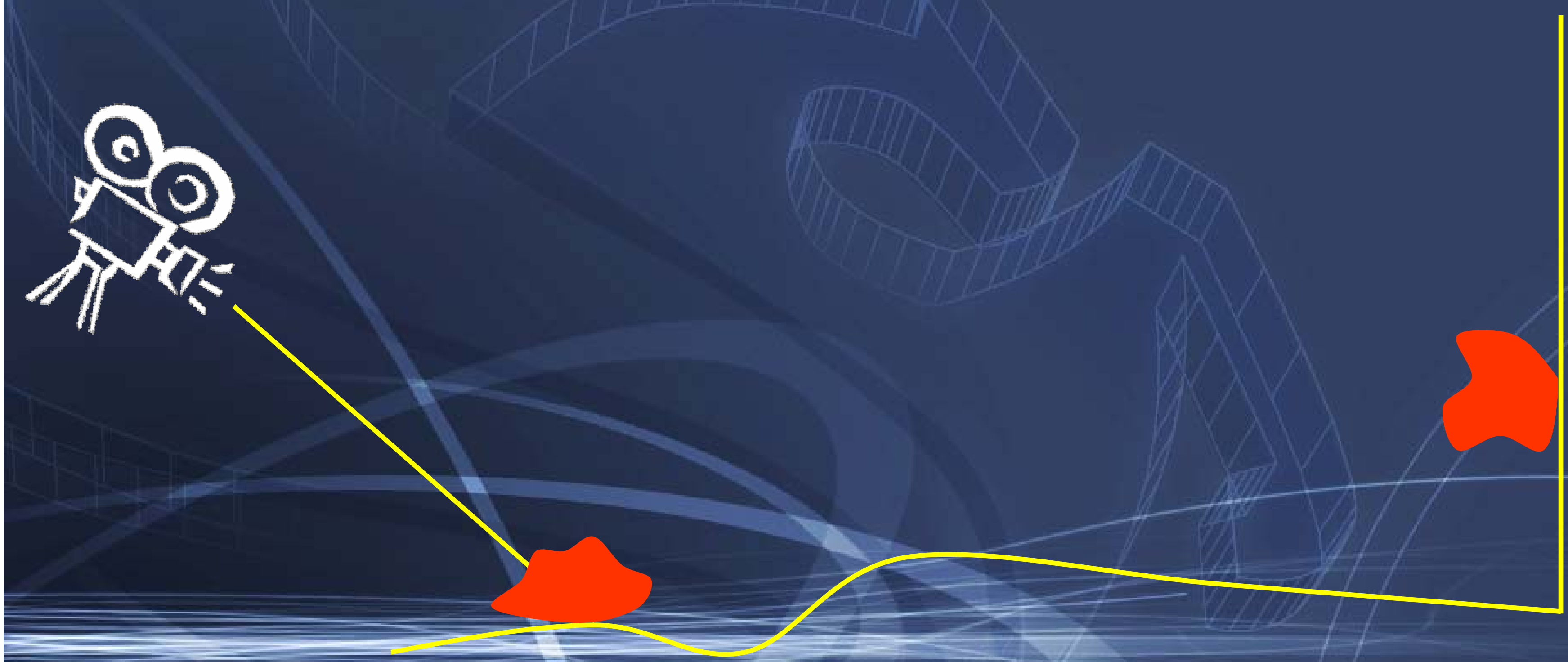
Rotation on CPU for SH and HSH

Added conversion (sparse matrix)

Accuracy overcomes computational overhead

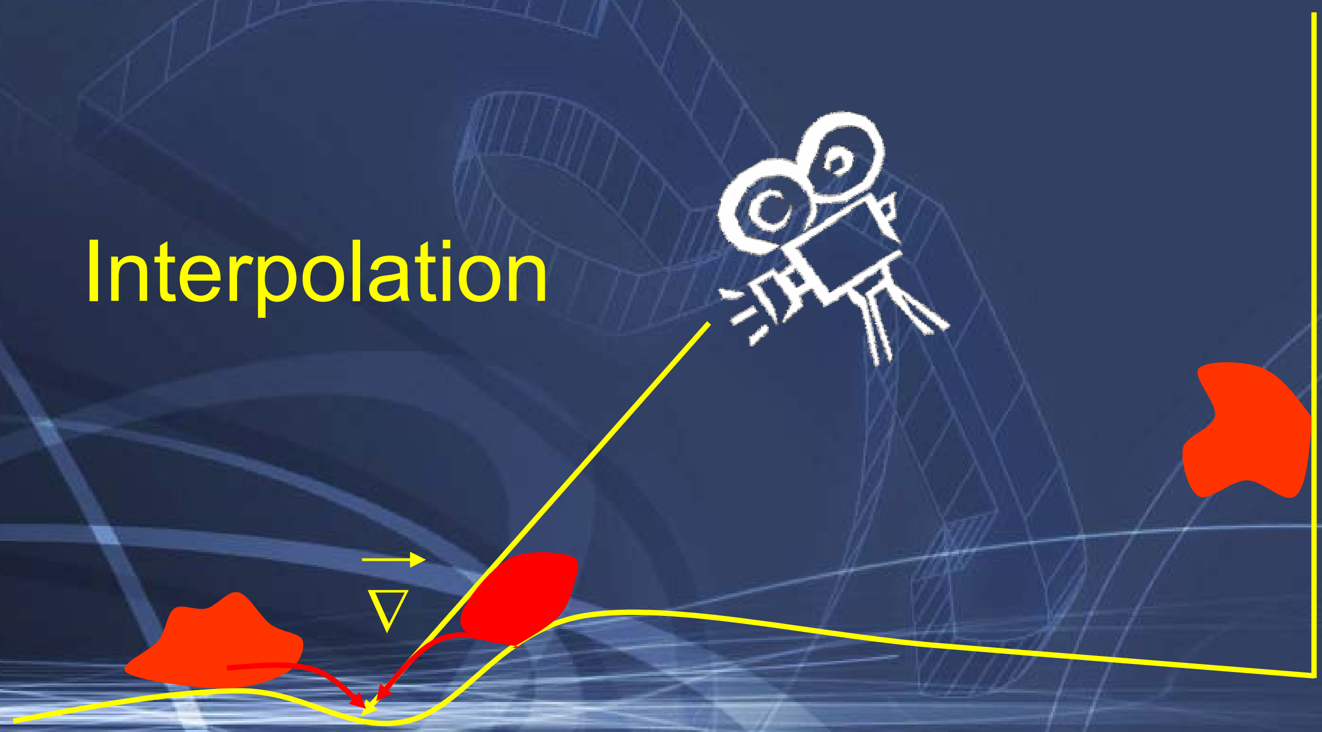
Application : Radiance Caching

Goal : computation of indirect **glossy** lighting



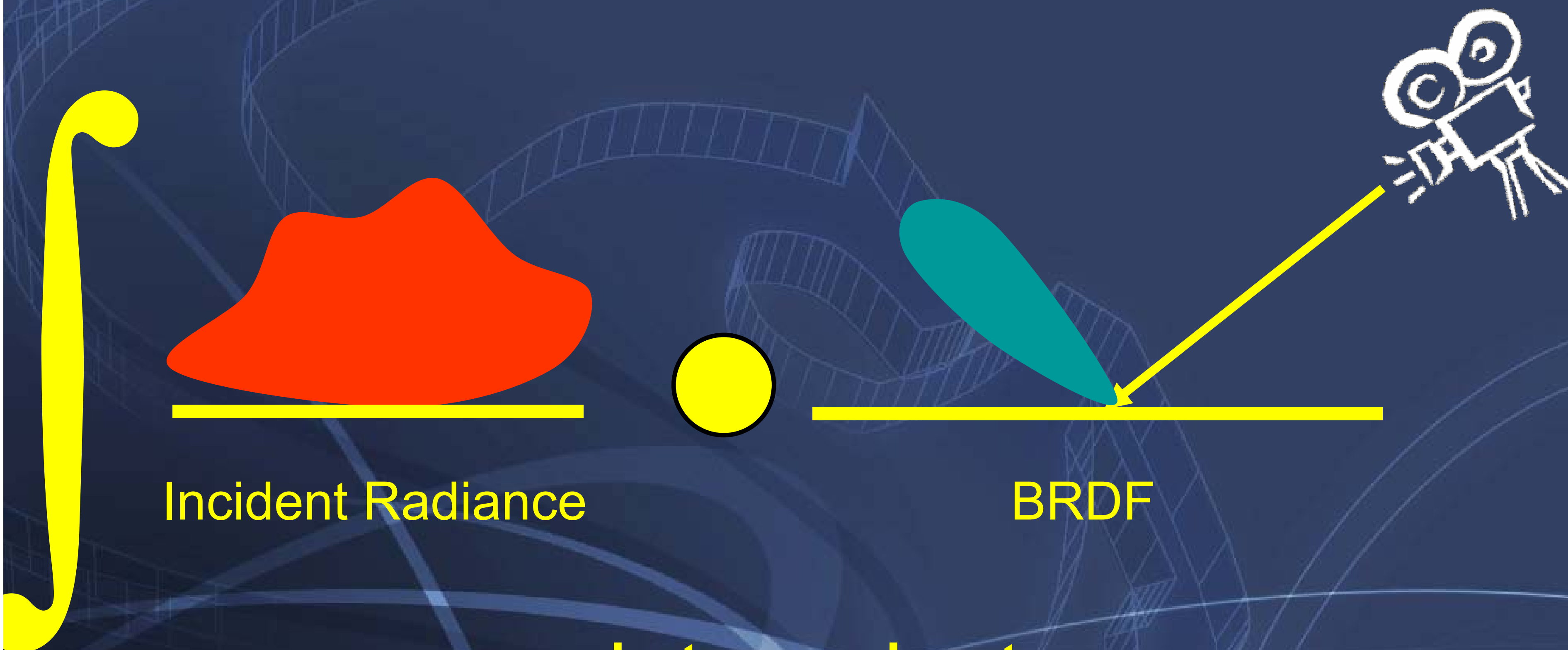
Application : Radiance Caching

Goal : computation of indirect **glossy** lighting



Application : Radiance Caching

Goal : computation of indirect **glossy** lighting



≡ dot product

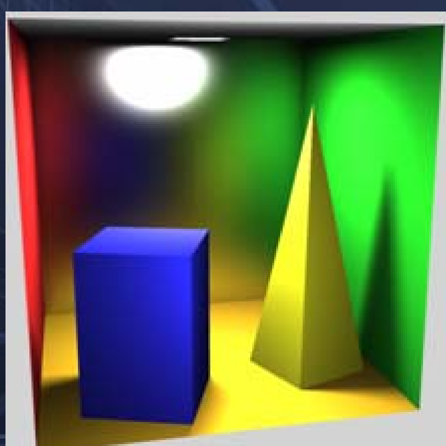
Application : Radiance Caching

Results

Low frequency BRDFs

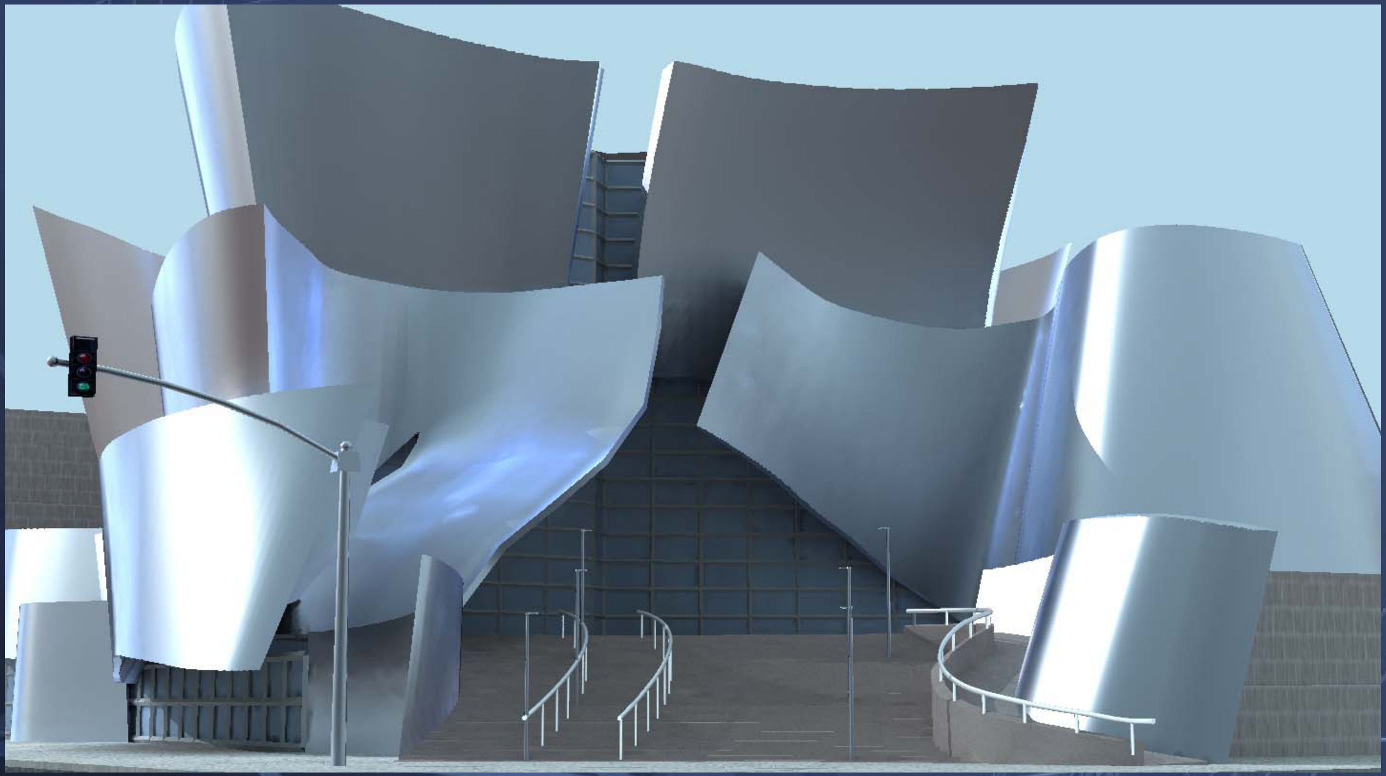
Rotational gradient
replaced by rotation

New translational
gradients formulas



Examples

Examples



Examples

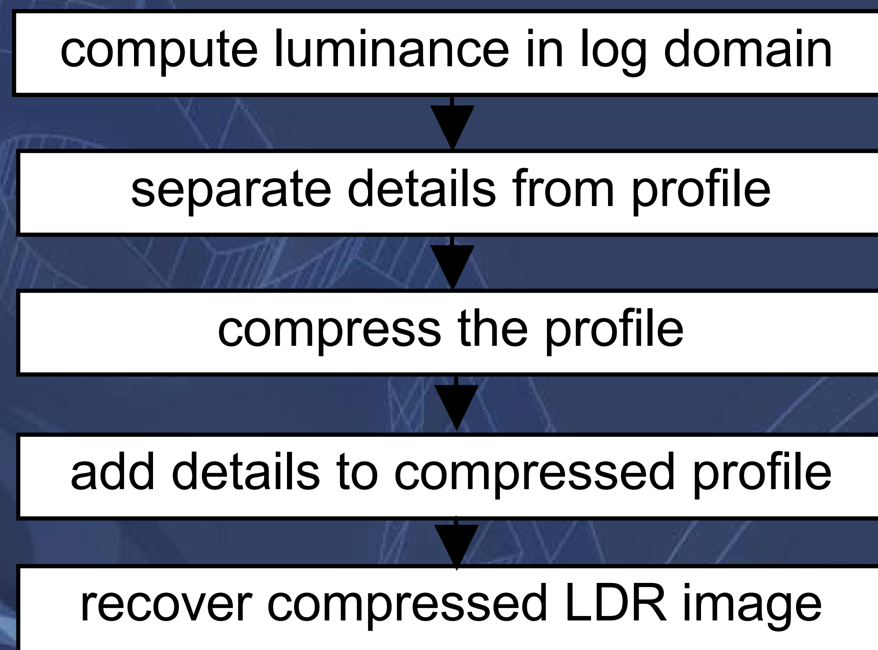
Examples

Rendering
Related Research

- Detail Extraction using Level Set Method for Realistic Display
- Monte Carlo Noise Reduction
- HDR Image Data Compression

Level Set Tone Mapping

- Separate into profile and detail
- Compress profile
- Add back detail



Level Set Tone Mapping

– Find profile using level set methods

$$I_t + F(\kappa)\left|\nabla I\right| = 0$$

- *I* is the luminance of HDR image
- *k* is the curvature
- *F(k)* is speed function

$$F(\kappa) = -e * \kappa$$

$$e = \exp\left(\frac{-\,sen *}{\sqrt{\left|D_x^+ - D_x^-\right|^2 + \left|D_y^+ - D_y^-\right|^2}} \right)$$

Level Set Tone Mapping

Comparison of images resulting from our method (right) vs. resulting from a method using S-compression alone (left).

Monte Carlo Noise Reduction

- Facts
  - insufficient sampling rate
  - Outliers & inter-pixel incoherence
- Actual pixel value contaminated
  - additive or multiplication?

Monte Carlo Noise Reduction

- Natural image statistics

$$P_c(c; s_c, p_c) = \frac{1}{Z} e^{-|c/s_c|^{p_c}}$$

$$Z_c = 2\frac{s_c}{p_c}\Gamma(\frac{1}{p_c})$$

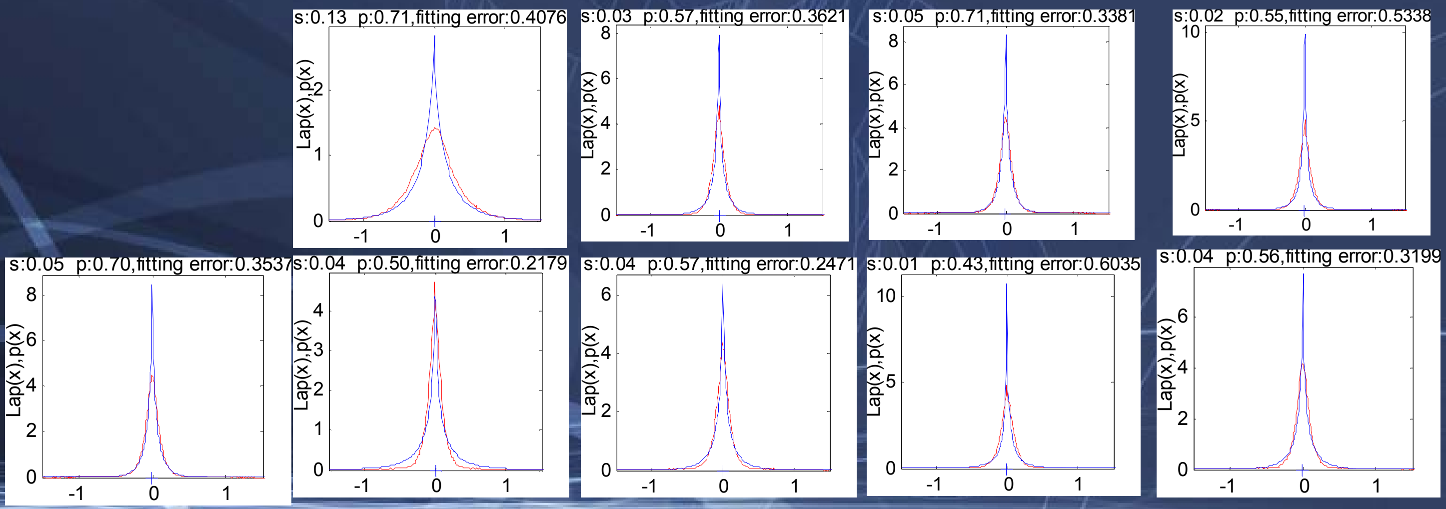
where s_c, p_c can be recovered by the following equations:

$$\sigma_y^2 = \frac{s_c^2\Gamma(3/p_c)}{\Gamma(1/p_c)} + \frac{s_n^2\Gamma(3/p_n)}{\Gamma(1/p_n)}$$

$$m_y^4 = \left(\begin{array}{l} \frac{s_c^4\Gamma(5/p_c)}{\Gamma(1/p_c)} + \frac{s_n^2\Gamma(5/p_n)}{\Gamma(1/p_n)} \\ + \frac{6s_c^2 s_n^2\Gamma(3/p_c)\Gamma(3/p_n)}{\Gamma(1/p_c)\Gamma(1/p_n)} \end{array} \right)$$

Monte Carlo Noise Reduction

- Statistical noise model
  - *Laplacian* modeling



Simoncelli, E. P., Freeman W. T., THE STEERABLE PYRAMID: A FLEXIBLE ARCHITECTURE FOR MULTI-SCALE DERIVATIVE COMPUTATION. IEEE ICIP' 1995.

Monte Carlo Noise Reduction

- Beyesian denoising
  - Maximum value
  - Expected value (we take this)

$$\hat{c}(y) = \int P_{c|y}(c \mid y) c\,dc$$

$$\hat{c}(y) = \frac{\int P_n(y-c) P_c(c) c\,dc}{\int P_n(y-c) P_c(c)\,dc}$$

  - *s, p are user specified parameters*
    - *S*: [0.0, 0.15]
    - *p*: [0.5, 1.0]

Monte Carlo Noise Reduction

- Results

Monte Carlo Noise Reduction

- Bilateral Filtering

$$h(x) = \frac{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\xi)c(\xi,x)s(f(\xi),f(x))d\xi}{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} c(\xi,x)s(f(\xi),f(x))d\xi}$$

$$c(\xi,x) = \exp\left\{-\frac{1}{2}\left(\frac{|\xi - x|}{\sigma_d}\right)^2\right\}$$

$$s(f(\xi),f(x)) = \exp\left\{-\frac{1}{2}\left(\frac{|f(\xi) - f(x)|}{\sigma_r}\right)^2\right\}$$

  – Adaptive filtering kernel: $f(x) \rightarrow \tilde{f}(x)$

$$\tilde{f}(x) = \frac{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\xi)c(\xi,x)d\xi}{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} c(\xi,x)d\xi}$$

Monte Carlo Noise Reduction

- Numerical formulation

$$\hat{f}(i,j)=\frac{\sum_{u=-3\sigma_d}^{3\sigma_d}\sum_{v=-3\sigma_d}^{3\sigma_d} f(i+u,j+v)c(u,v)s(f(i+u,j+v),\tilde{f}(i,j))}{\sum_{u=-3\sigma_d}^{3\sigma_d}\sum_{v=-3\sigma_d}^{3\sigma_d} c(u,v)s(f(i+u,j+v),\tilde{f}(i,j))}$$

$$\tilde{f}(i,j)=\frac{\sum_{u=-3\sigma_d}^{3\sigma_d}\sum_{v=-3\sigma_d}^{3\sigma_d} f(i+u,j+v)c(u,v)}{\sum_{u=-3\sigma_d}^{3\sigma_d}\sum_{v=-3\sigma_d}^{3\sigma_d} c(u,v)}$$

- window size: $6\sigma_d \times 6\sigma_d$
- parameters: $\sigma_r = 2, \sigma_d = 0.4$

Monte Carlo Noise Reduction

- results

	noisy	denoised	accurate	
Living room 400×300	50s(2) 0.152	5.0s 0.089	2100s (500)	2, 0.4
Cabin 512×512	286s(20) .3630	9.8s .2202	3602s (300)	2, 0.4
Conf. room 512×347	183s(5) .0312	6.5s .0275	1802s (400)	2, 0.4

noisy, 20 samples

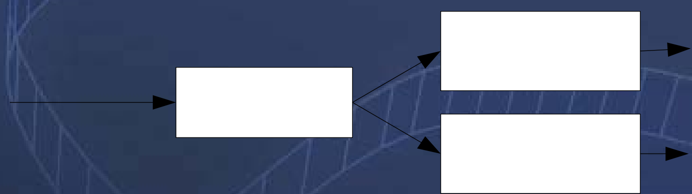
left: our method
middle: standard bilateral filtering
right: Wiener filtering

HDR Image Compression

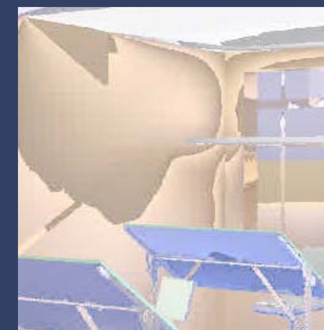
- 3 floats for R,G,B
  - 12 bytes / pixel
- High dynamic range
  - existing image/video codec not applicable
    - 3 bytes for R,G,B

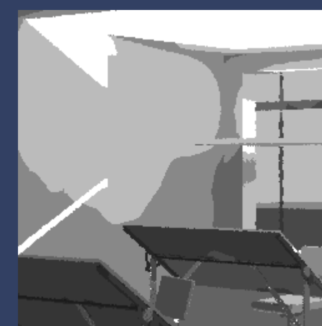
HDR Image Compression

- DCT Based
  - Compression scheme



$$
\begin{cases}
v = \max(r, g, b) \\
m, e : \text{mantissa, exponent of } v \\
V = 256 \cdot m / v \\
R = r \cdot V \\
G = g \cdot V \\
B = b \cdot V \\
E = e + 128
\end{cases}
$$





  - HDR image lossless compression:
    - Color base: lossless (CABAC [FFV1], LJPEG, etc.)
    - Common exponent: lossless (CABAC)
  - HDR image lossy compression:
    - Color base: lossy (AJPEG, etc.)
    - Common exponent: lossless (CABAC)

HDR Image Compression

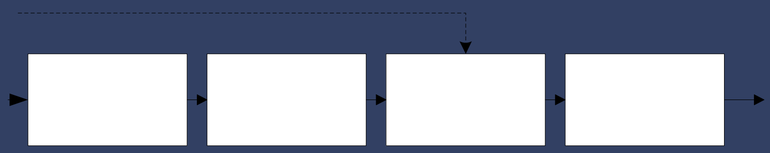
- Lossy RGB compression
  - *E is used to guarantee minimum quantization error for each block pixel.*
  - Adaptive quantization

$$Q = Q/q_c$$

$$q_c = 2^{\max\{E_i\}-\min\{E_i\}}, i = 1,\ldots,64$$

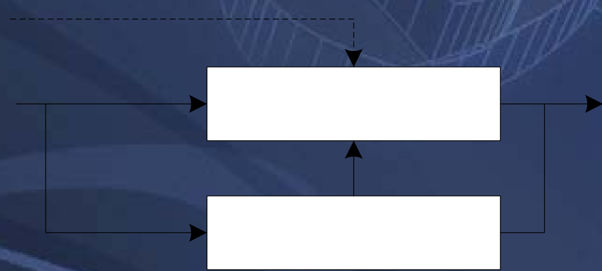
or

$$q_c = \left\lfloor \frac{\max\{E_i\}-\min\{E_i\}+3}{2} \right\rfloor$$



HDR Image Compression

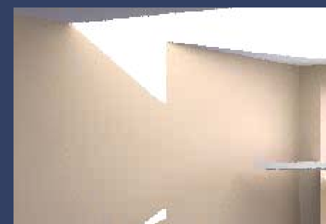
- HDR video compression
  - HDR video lossless compression
    - Color base: lossless (CABAC, etc.)
    - Common exponent: lossless (CABAC)
  - HDR video lossy compression
    - Color base: lossy (MJPEG, MPEG, etc.)
    - Common exponent: lossless (CABAC)

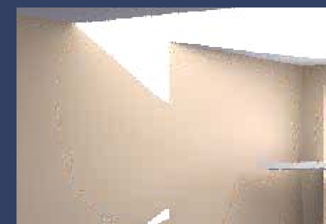


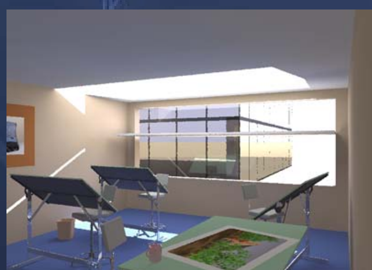
HDR Image Compression

- "room"

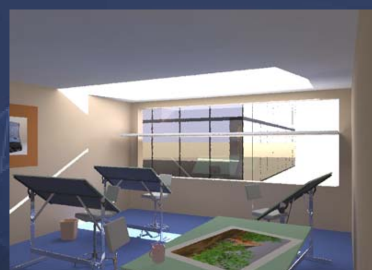
  Size: 800X754 dynamic range: [0.001, 20.875]



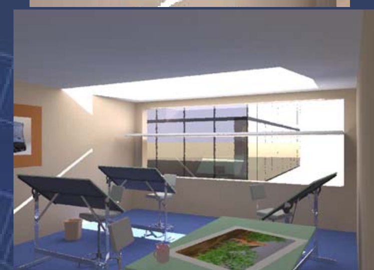




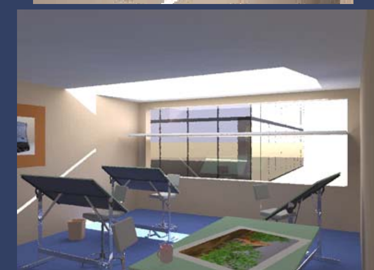
(a) Original



(b) AJPEG+CABAC, 0



(c) AJPEG+CABAC, 16

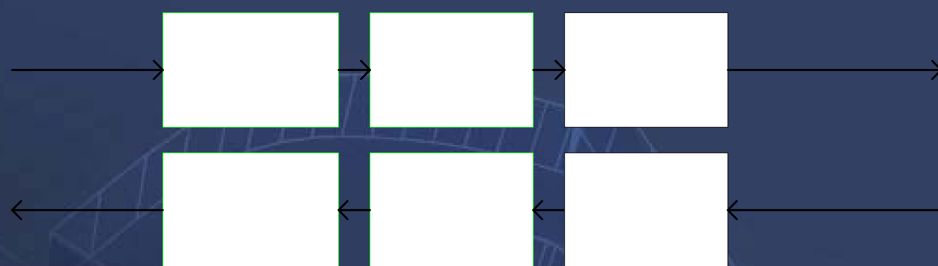


(d) JPEG+CABAC, 16

	Size	t_c	t_d	*MSE*	*PSNR*	*VDP*
	KB	*sec.*	*sec.*	10^{-3}	*dB*	10^{-3}
RGBE	786.0,69.0%	n/a	n/a	n/a	n/a	n/a
OpenExr	701.0, 71.0%	n/a	n/a	n/a	n/a	n/a
CABAC+CABAC	181.0, 92.5%	0.36	0.130	n/a	n/a	n/a
LS-JPEG+CABAC	269.0, 89.0%	0.30	0.090	n/a	n/a	n/a
AJPEG+CABAC,0	123.0, 95.0%	0.38	0.047	0.13	65.2	.59
AJPEG+CABAC,16	78.5, 96.8%	0.41	0.046	0.62	58.5	1.1
JPEG+CABAC, 16	51.0, 97.9%	0.38	0.046	1.85	53.7	1.8

HDR Image Compression

- JPEG 2000 based



HDR Image Compression/Decompression Scheme

HDR Image Encoder: Raw HDR Image Data | Logarithm Transform | Quantization

HDR Image Decoder: Raw HDR Image Data | Inverse Logarithm | Inverse Quantization

HDR Image Compression

- JPEG 2000 based
  - Logarithm transform and quantization

$$[r', g', b'] = \log([r, g, b])$$

$$[\bar{r}, \bar{g}, \bar{b}] = f([r', g', b'] : n)$$

$$f(x : y) = \left\lfloor \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (2^y - 1) \right\rceil$$

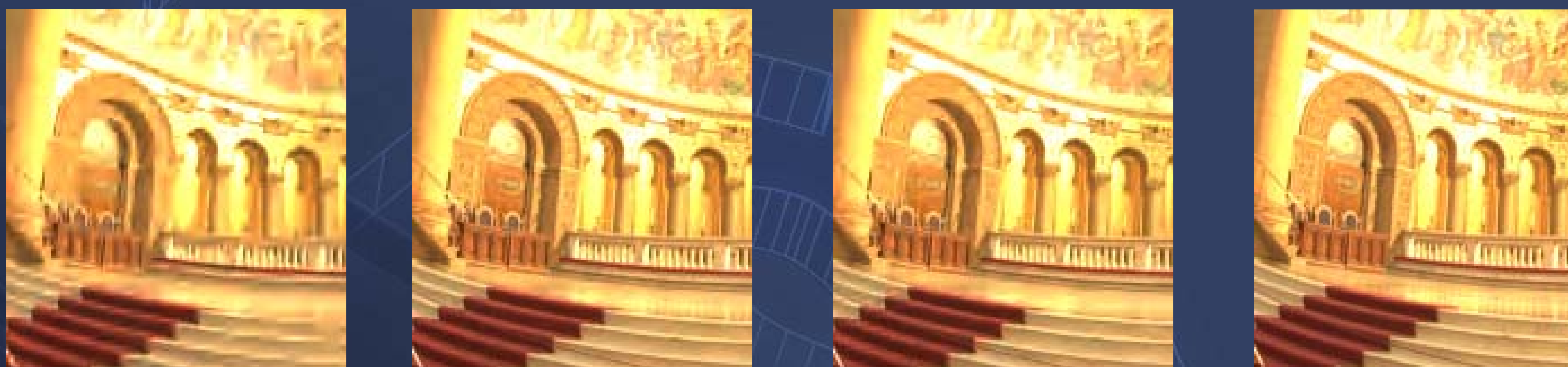
  - Quantization error

$$\varepsilon = \frac{x_{\max} - x_{\min}}{2^{n+1} - 2}$$

HDR Image Compression

- **“memorial”**

size 512X768, 1,313K(RGBE), 823K(OpenExr)



0.01 0.02 0.03 reference

rate	**0.01**	**0.02**	**0.03**
Size(KB)	23	46	70
MSE	0.60	0.20	0.094
VDP(10^{-3})	96	69	53
Timing(s.)	1.7/0.64	1.7/0.68	1.7/0.67

Compression rate Comparison

HDR Image Compression

reference compressed

	rate=0.03	RGBE	OpenExr
"park"	65	1,116	823
"memorial"	70	1,313	1,253
"BigFogMap"	149	2,683	2,178
"tahoe"	553	9,668	7,172

Comparison with other formats

Cache Based Real_time Rendering

Minimally Computing Cache and Hardware Splatting for faster rendering

- Cache Point Computation
  - Empirical 3D Space Sampling
    - Scene complexity metric
      - Geometric complexity
      - Neighborhood complexity

Students supported

- Ruifeng Xu
  - High Dynamic Range Compression and Display
  - Physically Based Rendering
  - MR rendering
- Jaakko Konttinen
  - Real Time Rendering
  - MR Rendering
  - Physically Based Rendering
- Mark Colbert
  - Rendering Interface
  - Physically Based Rendering
- Ahmet O. Akyuz
  - HDR Imaging
  - Color Transfer
- Yugang Min
  - MR in PDA
- Pascal Gautron (from IRISA/INRIA France)
  - Physically Based Rendering
  - Real-Time Rendering
- Jaroslav Krivanek (from IRISA/INRIA France)
  - Physically Based Rendering
  - Real-Time Rendering
  - Perception Based Rendering

Publications

- C. E. Hughes, Jaakko Konttinen and S. N. Pattanaik, “The Future of Mixed Reality: Issues in Illumination and Shadows,” *Proceedings of I/ITSEC 2004*, Orlando, December 6-9, 2004.
- C. E. Hughes, Erik Reinhard, Jaakko Konttinen and S. N. Pattanaik, “Achieving Interactive-Time Realistic Illumination in Mixed reality,” *Proceedings of Army Science Conference (ASC) 2004*, Orlando, FL, November 29-December 2, 2004. (Poster Presentation Only)
- Erik Reinhard, Ahmet O. Akyuz, Mark Colbert, Matthew O’Connor and C. E. Hughes, “Real-Time Color Blending of Rendered and Captured Video,” *Proceedings of I/ITSEC 2004*, Orlando, December 6-9, 2004.
- Ruifeng Xu, S. N. Pattanaik and C. E. Hughes, “Real-time Rendering of Dynamic Objects in Dynamic, Low-Frequency Lighting Environments,” *Computer Animation and Social Agents (CASA 2004)*, Geneva, Switzerland, July 7-9, 2004.

Publications (cont..)

- Oguz Ahmet, Erik Reinhard, S. N. Pattanaik, “Color Appearance in High Dynamic Range Imaging”, Conference on Applied Perception 2004 (Accepted, to be presented in Aug 2004).
- Ruifeng Xu, S. N. Pattanaik, “Non-Iterative, Robust Monte Carlo Noise Reduction”, *IEEE Computer Graphics and Applications*. Accepted for publication. February 2004.
- Pascal Guatron, Jaroslav Krivakek, S. N. Pattanaik, Kadi Boutaouch, “A Novel Hemispherical Basis for Accurate and Efficient Rendering”. Proceedings of 2nd *Eurographics Symposium on Rendering* 2004 (July 2004).
- Ruifeng Xu, S. N. Pattanaik, Charles Hughes, “Real-time rendering of dynamic objects in dynamic, low frequency environments”. Proceedings of *Computer Graphics and Social Agents Conference* (CASA) 2004, MIRALab, Switzerland, Accepted in May 2004.

Publications (cont..)

- Jaroslav Krivakek, S. N. Pattanaik, Jiri Zara, “Adaptive Mesh Subdivision for Precomputed Radiance Transfer”, *Proceedings of Spring Conference in Computer Graphics* (SCCG 2004), Slovak Republic, ACM-Press, 2004.
- J. Krivakek, P. Gautron, S. N. Pattanaik, K. Bouatouch, “Radiance Caching for Efficient Global Illumination Computation”, Accepted in TVCG (Jan 2004).
- Walter Mundt, S. N. Pattanaik, Erik Reinhard, “Beyond Triangles: Using Ray-Casting to Render New Directly Primitives In Graphics Hardware” accepted for GDC 2005 Poster presentation.

Submissions (under review)

- Jaroslav Krivakek, Jaakko Konttinen, S. N. Pattanaik, K. Bouatouch, "Fast Approximation to Spherical Harmonic Rotation", Submitted for publication in Journal of Graphics Tools (2005).
- Ruifeng Xu and S. N. Pattanaik, C. E. Hughes "HDR Still Image in JPEG 2000", submitted for publication in Journal of Graphics Tools (2004).
- Ruifeng Xu and S. N. Pattanaik, C. E. Hughes "High Dynamic Range Image and Video Data Compression", submitted for publication in Visual Computers (2004).

Tutorials Presentations

- Paul Debevec, Erik Reinhard, Greg Ward, Sumanta Pattanaik , “High Dynamic Range Imaging”, ACM-SIGGRAPH 2004.
- Sumanta Pattanaik, Erik Reinhard, “High Dynamic Range Imaging” Game Developer Conference, March 2004.