Illumination Capture and Rendering for Augmented Reality

Simon Gibson
University of Manchester, UK

RFIA 2004
Augmented Reality

• *The process of augmenting a view of a real scene with additional objects or information*

• Some applications require visually realistic synthetic objects
  – Interior design
  – Broadcast
  – Computer Games/Entertainment
  – …
Visual realism

• Geometric
  – Register object correctly with the camera
  – Resolve occlusions between real and virtual objects

• Radiometric
  – Shade the object consistently with the rest of the scene
  – Cast shadows between real and virtual objects
Tutorial overview

• Modelling and rendering for Augmented Reality
• Geometric camera calibration
• Modelling scene geometry
• HDR Illumination reconstruction
• Rendering and compositing
  – Shading and shadows
• Examples and Videos
Modelling and rendering for Augmented Reality

• Basic requirements to augment images
  – Camera position for current frame
  – 3D scene model to resolve occlusions and collisions
  – HDR representation of the scene lighting
  – Radiometric camera response
Modelling and rendering for Augmented Reality

- Calibrate the camera position and model scene geometry using interactive techniques
- Capture illumination using a light-probe and HDR imaging, storing data with scene model
Modelling and rendering for Augmented Reality

- Rendering objects using OpenGL
  - Clear colour and depth buffers
  - Draw background image into colour buffer
  - Set OpenGL camera state to match calibrated camera position
  - Draw 3D scene model into depth buffer
  - Draw virtual objects into colour and depth buffers
Modelling and rendering for Augmented Reality

- “Illumination mesh” used to shade virtual objects
  - Diffuse & specular reflection
  - Multi-pass rendering
  - GPU shadow mapping
- Virtual objects rendered at interactive rates (5-15fps)
Tutorial overview

• Modelling and rendering for Augmented Reality
• Geometric camera calibration
• Modelling scene geometry
• HDR Illumination reconstruction
• Rendering and compositing
  – Shading and shadows
• Examples and Videos
To register objects correctly with the real camera, we need to know the camera projection matrix, \( P \):

\[
P = K.[R|t]
\]

where

- \( R \) = camera rotation (3x3 matrix)
- \( t \) = camera translation (3x1 vector)
- \( K \) = camera calibration parameters (3x3 matrix)
Geometric camera calibration

- Moving camera
  - Must be tracked for each frame
  - Intrinsic parameters can generally be calculated off-line
  - Extrinsic parameters updated at each frame using marker-based or marker-less tracking algorithms

- Static camera
  - Can be calibrated off-line
  - 2 or more orthogonal “vanishing points” provide enough information to determine both intrinsic and extrinsic parameters
Static camera calibration

User draws two or more orthogonal vanishing points in the image
Tutorial overview

• Modelling and rendering for Augmented Reality
• Geometric camera calibration
• *Modelling scene geometry*
• HDR Illumination reconstruction
• Rendering and compositing
  – Shading and shadows
• Examples and Videos
Reconstructing scene geometry

• 3D geometric model needed for:
  – Resolving occlusions between real and virtual objects
  – Collision detection with background scene
  – Constructing an illumination environment to shade objects
  – “Catching” shadows cast by virtual objects
Reconstructing scene geometry

• Many different approaches…
• Automatic
  – Dense stereo matching (e.g. Pollefeys ‘01)
  – Laser scanning (e.g. Levoy ‘00)
  – Structured light projection (e.g. Scharstein ‘03)
• Semi-automatic
  – Primitive-based modelling (e.g. Debevec ‘96, Cipolla ‘98, El Hakim ‘00, Gibson ‘03)
Automatic scene reconstruction

Dense stereo matching requires multiple images and scene texture
Automatic vs. semi-automatic reconstruction

**Automatic**
- ✓ Accurate geometry
- ✓ Relatively fast and easy-to-use
- ✗ Unstructured geometry
- ✗ Hole and occlusion problems
- ✗ Requires scene “texture”
- ✗ Requires more than one image

**Semi-automatic**
- ✗ Less accuracy & simple shapes
- ✗ More labour-intensive
- ✓ Good scene structure
- ✓ Can apply “user-knowledge”
- ✓ Works without scene “texture”
- ✓ Works with a single image
Automatic vs. semi-automatic reconstruction

**Automatic**
- ✓ Accurate geometry
- ✓ Relatively fast and easy-to-use
- ✗ Unstructured geometry
- ✗ Hole and occlusion problems
- ✗ Requires scene “texture”
- ✗ Requires more than one image

**Semi-automatic**
- ✗ Less accuracy & simple shapes
- ✗ More labour-intensive
- ✓ Good scene structure
- ✓ Can apply “user-knowledge”
- ✓ Works without scene “texture”
- ✓ Works with a single image

*For Augmented Reality, we don’t need a “pixel perfect” scene model!*
Semi-automatic geometry reconstruction

• Manipulate geometric primitives to match features in the calibrated background image
  – Use a library of pre-defined geometric shapes
  – User-specified constraints assist in the modelling process
  – Non-linear optimisation algorithm adjusts primitives so constraints are satisfied

(see Gibson et al. Computers and Graphics, April 2003)
Semi-automatic geometry reconstruction
Semi-automatic geometry reconstruction

~20 mins construction time
Example scene reconstruction
Tutorial overview

• Modelling and rendering for Augmented Reality
• Geometric camera calibration
• Modelling scene geometry
• **HDR Illumination reconstruction**
• Rendering and compositing
  – Shading and shadows
• Examples and Videos
Illumination reconstruction

• In order to illuminate synthetic objects, we must build a representation of the real light in the scene
• Capture an *image-based* representation of light using a “light-probe” and calibration grid
High dynamic range imaging

- Standard cameras are not able to capture the full dynamic range of light in a single image

Bright areas are washed-out

Loss of detail in dark areas
High dynamic range imaging

- By combining multiple images at different exposure times, all detail can be recovered, as well as the camera response function.

Short exposure times give detail in bright areas.

Long exposure times give detail in dark areas.
Recovering camera response

See Debevec and Malik, SIGGRAPH 1997 for further details
Calibrating the probe image

- Known spot locations can be used to find the camera projection matrix.
- Known light-probe dimensions used to locate probe area in HDR image
  - Probe always positioned over most distant spot
  - Probe size and mount height known
Calibrating the probe image

- Light-probe is not a perfect reflector
- Can approximate light-probe reflectivity using pixel ratios in HDR image

Reflectivity ≈ (0.53, 0.50, 0.48)
Orienting the light probe within the scene

• In order to map radiance onto scene, light-probe must be correctly positioned and oriented within the scene model.

• Reconstruct geometric representation of calibration grid, and use to position and orient light-probe coordinate system.
Orienting the light probe within the scene
Building the illuminated scene model

- Radiance values projected outwards from the light-probe location onto the scene model
- Lighting information stored in a triangular “illumination mesh”
- Radiosity transfer used to estimate missing values
Building the illuminated scene model

- Diffuse reflectance for each visible patch estimated using the ratio of radiance to irradiance:
  \[
  \text{reflectance} \approx \frac{L_i}{E_i}
  \]
- Radiance obtained from HDR image
- Irradiance gathered from environment using geometric form-factors
Tutorial overview

• Modelling and rendering for Augmented Reality
• Geometric camera calibration
• Modelling scene geometry
• HDR Illumination reconstruction
• Rendering and compositing
  – Shading and shadows
• Examples and Videos
Illuminating synthetic objects

- Traditionally done using raytracing (e.g. Debevec ’98)
  - High accuracy
  - Supports a wide range of surface reflectance functions
  - Not interactive

- Recent advances in GPU functionality allow interactive rendering with image-based lighting (e.g. Sloan ’03)
  - Fast rendering
  - Supports a wide range of surface reflectance functions
  - Only supports distant illumination
Illuminating synthetic objects

• Render diffuse and specular reflectance separately
• Diffuse shading uses an *Irradiance Volume* (Greger ’98)
• Specular reflections approximated using dynamically generated environment maps

✓ Fast and easy to use
✓ Supports non-distant illumination
✗ Limited support for complex reflectance functions
**Irradiance volume**

- Volumetric representation of irradiance at every point and direction in space
- Irradiance sampled at every grid vertex and store in the volume
- *Spherical Harmonics* used to reduce storage and computation requirements
- Vertex shaded by querying irradiance using table lookups and interpolation

*See Greger et al, IEEE CG&A 1998, and Ramamoorthi and Hanrahan, SIGGRAPH 2001*
Querying the irradiance volume

- To shade each object vertex, we query the irradiance volume using vertex position and normal direction
- Irradiance at voxel corners evaluated by summing spherical harmonic coefficients
- Irradiance evaluated at vertex position using tri-linear interpolation
Diffuse shading

- Irradiance at each vertex is multiplied by the object’s diffuse reflectance
- Radiosity is tone-mapped for display using the camera response function
- Object is drawn using tone-mapped vertex colours, and GPU is used to interpolate shading across each triangle
Example of diffuse shading

• Can shade approximately 350K vertices per second on a 2.5GHz Pentium4
• Equivalent to 11K vertices at 30 frames-per-second
Specular reflections

• Specular reflections generated using dynamically generated cubic environment maps

• We assume that the diffuse and specular components can be tone-mapped independently and combined:

\[ T(D+S) \approx T(D) + T(S) \]
Specular reflections

- Multiply 3D scene radiance by object’s specular reflection coefficient
- Tone-map scene using table lookup into camera response function
- Render cubic environment maps
- Render virtual object again, with environment mapping enabled
Specular reflections

- Diffuse and specular components combined using OpenGL additive blending
- We assume:
  \[ T(D+S) \approx T(D) + T(S) \]
Rendering shadows

- We must account for four types of shadow:
  - Virtual-to-real (*shadow cast by synthetic object*)
  - Virtual-to-virtual (*self-shadowing on synthetic object*)
  - Real-to-virtual (*shadow cast onto synthetic object*)
  - Real-to-real (*existing shadow in the real scene*)
Rendering shadows

• We must account for four types of shadow:

  – Virtual-to-real (shadow cast by synthetic object)
  – Virtual-to-virtual (self-shadowing on synthetic object)

  – Real-to-virtual (shadow cast onto synthetic object)
  – Real-to-real (existing shadow in the real scene)

\[
\begin{align*}
\text{Irradiance volume} & \quad \text{Background image}
\end{align*}
\]
Overview of shadow rendering

1. Build a shaft hierarchy to encode light transport between all patches in the scene model
2. Identify the significant sources of shadow for each virtual object
3. Generate a shadow map from each shadow source
4. Remove contributions of light emitted by each source from the background image

Repeat for every frame
Related work

• Soft-shadow approximated using multiple overlapping hard shadows
  – Brotman and Badler ’84, Herf and Heckbert ’97

• Shadow generation for Augmented Reality

• Line-space hierarchy to encode light transport
  – Drettakis and Sillion ’97

• HDR Image-based lighting and differential rendering
  – Debevec ’98, Sato ’99
Shaft hierarchy

- Shaft hierarchy stores a coarse representation of the light transport from groups of source patches to groups of receiver patches.

- Assume source and receiver patches emit and reflect light diffusely.

- Radiance transfer is pre-computed between each source and receiver pair.
Radiance transfer pre-computation

- Calculate “form-factor” between each source patch and each receiver
- Visibility evaluated using ray-casting
- Radiance transfer summed and stored within shaft hierarchy
Shadow source identification

• Given the bounding box of an object, a list of potentially occluded shafts can be found very quickly before rendering each frame.

• Define source set:
  – All source patches associated with all potentially occluded shafts.

• Define receiver set:
  – All receiver patches associated with all potentially occluded shafts.
Differential rendering

- Given a rendered image, $I_{\text{obj}}$, of scene and virtual objects and a second image $I_{\text{noobj}}$ without virtual objects, subtract their difference, $I_e$ from the background photograph:

$$I_{\text{final}} = I_b - I_e = I_b - (I_{\text{noobj}} - I_{\text{obj}})$$
Shadow rendering

- Assuming we can operate entirely with HDR frame buffer data, repeat for each patch in the source set:
  1. Generate a shadow map that encloses the synthetic object
  2. Enable OpenGL subtractive blending and initialise projection/modelview matrices to the calibrated camera position
  3. Render receiver set with shadow mapping enabled, with vertex colours set to the pre-computed radiance transfer from the source patch
HDR image operations

• Ideally, we want to perform all operations on HDR image data and tone-map afterwards

\[ I_{\text{final}} = T(L_b - \sum L_j M_j) \]

- \( L_b \) is the HDR background image
- \( L_j \) is the radiance transfer from source patch j
- \( M_j \) is the shadow mask (1 occluded, 0 unoccluded)
- \( T() \) is the camera response function
Mapping to LDR image data

• If \( T() \) was linear, we could separate the terms:

\[
I_{\text{final}} = T(L_b) - T(\sum L_j M_j)
\]

\[
= I_b - \sum M_j T(L_j)
\]
Mapping to LDR image data

• If $T()$ was linear, we could separate the terms:

$$I_{\text{final}} = T(L_b) - T(\sum L_j M_j)$$

$$= I_b - \sum M_j T(L_j)$$

• Camera response is non-linear, so $T(A-B) \neq T(A) - T(B)$

$$I_{\text{final}} = I_b - \sum S_j M_j$$

$S_j$ is LDR “equivalent” of $L_j$
Mapping to LDR image data

• Non-linearity of $T()$ means that we can only approximate $S_j$
  – Dependent on order of evaluation of source patches
  – Requires knowing all $M_j$ before evaluation
Mapping to LDR image data

- Non-linearity of $T()$ means that we can only approximate $S_j$
  - Dependent on order of evaluation of source patches
  - Requires knowing all $M_j$ before evaluation

- Fix order of source patch evaluation (brightest to darkest)
- Assign approximate $M_j$ (e.g. 50% occluded, 50% unoccluded)

*(see Gibson et al. Eurographics Rendering Symposium, 2003)*
Examples of shadow rendering
Examples of shadow rendering
Self-shadows

- Virtual objects cast shadows onto themselves!
- Standard shadow-mapping can’t be used, because we can’t pre-compute radiance transfer to the virtual object
- Pre-compute an approximate per-vertex visibility term to account for self-occlusions
Ambient Occlusion

- Commonly used in film post-production to approximate global-illumination effects
- Assumes that object is illuminated by a uniform sphere of light

Rapid Shadow Generation in Real-World Lighting Environments

Simon Gibson, Jon Cook
Toby Howard and Roger Hubbold
Visual comparison (1)

Interior, natural illumination

14 frames-per-second  Ray-traced  Photograph
Visual comparison (2)

Interior, artificial illumination

12 frames-per-second  Ray-traced  Photograph
Visual comparison (3)

Exterior, natural illumination

16 frames-per-second  Ray-traced  Photograph
Visual Comparison (4): Self-Shadows

No self-shadows (12fps)  With self-shadows (12fps)  Ray-traced
Summary

• Illumination capture achieved using HDR imaging and a light-probe

• Interactive rendering achieved using a combination of software and hardware techniques
  – Diffuse and specular shading
  – Soft shadows

• Rendering rates of between 5 and 15 frames-per-second on a GeForce4 Ti4600
Thank You!

Questions?