REAL-TIME MULTIPLE SCATTERING USING LIGHT PROPAGATION VOLUMES

Markus Billeter, Erik Sintorn & Ulf Assarson

Division of Computer Engineering
Chalmers University of Technology

i3D 2012 // Orange County, CA
Introduction - Why?

(Overview, no lighting)
Introduction - Why?

(Shadow Maps Only)
Introduction - Why?

With Volumetric Shadows
Introduction - Why?

With multiple scattering and related effects
Introduction - Why

• Volumetric Shadows (single scattering)
  – allows viewer to see where light comes from
  – no new surfaces illuminated
Introduction - Why

• Volumetric Shadows (single scattering)
  – allows viewer to see where light comes from
  – no new surfaces illuminated

• New method:
  – indirect illumination close to light “beams”
  – multiple scattering visible in medium
Introduction - Why

• Volumetric Shadows (single scattering)
  – allows viewer to see where light comes from
  – no new surfaces illuminated

• New method:
  – indirect illumination close to light “beams”
  – multiple scattering visible in medium

• Consider currently only homogenous and isotropic media
Introduction - How

- Render single scattering separately
  - high frequency
  - good methods exist
Introduction - How

• Render single scattering separately
  – high frequency
  – good methods exist

• Compute multiple scattering
  – lower frequency
  – use light propagation volumes
  – must consider higher-order scattering only
Introduction - How

• Render single scattering separately
  – high frequency
  – good methods exist

• Compute multiple scattering
  – lower frequency
  – use light propagation volumes
  – must consider higher-order scattering only

• Render multiple scattering effects
  – illuminate geometry in scene
  – capture in medium
Overview – Related Work

• Introduction
  – Why & How

• Related Work
  – Single Scattering
  – Light Propagation Volumes

• Propagation Scheme

• Implementation

• Results & Conclusion
Related Work – Single Scattering

- Single bounce
  - light to medium to eye
- “Airlight” Equation
  - for point light
  - solvable in shaders
  - homogenous and isotropic media
Related Work – Volumetric Shadows

- Evaluate Airlight in directly illuminated subspace only
- Found from shadow maps
- See “Real-Time Volumetric Shadows using Polygonal Light Volumes”
Related Work – LPV

- Light Propagation Volumes
  - “Cascaded Light Propagation Volumes for Real-Time Indirect Illumination”, Kaplanyan and Dachsbacher
  - Subdivide scene into cells/a grid
  - “Move light from one cell to it’s neighbors”
  - Fuzzy blocking

- Extended to consider scattering during propagation
Related Work – LPV

- Propagate radiance $L_c$:
  - from cell $c$
  - to neighboring cells $c'$
Related Work – LPV

• Propagate radiance $L_c$:
  – from cell $c$
  – to neighboring cells $c'_i$

• Use transfer functions $\Gamma_i$
Related Work – LPV

• Propagate radiance $L_c$:
  – from cell $c$
  – to neighboring cells $c'$

• Use transfer functions $\Gamma_i$

• Project radiance from $c$ on transfer functions
Related Work – LPV

- Propagate radiance $L_c$:
  - from cell $c$
  - to neighboring cells $c'$
- Use transfer functions $\Gamma_i$
- Project radiance from $c$ on transfer functions

\[ \Delta L_{c_6} = \langle L_c | \Gamma_6 \rangle \Gamma_6 \]
Overview – Propagation Scheme

• Introduction
  – Why & How

• Related Work
  – Single Scattering
  – Light Propagation Volumes

• Propagation Scheme
• Implementation
• Results & Conclusion
Propagation Scheme - Intro

• LPV: “just move light to neighboring cells”
• Want to include
  – extinction (absorption + out-scattering)
  – in-scattering
Propagation Scheme - Extinction

- Extinction reduces radiance moving through a cell
- Beer’s Law
  \[ I(d) = I_0 e^{-\beta d} \]
Propagation Scheme – In-scattering

• In-scattering:
  – compute total out-scattered radiance in cell
  – “redistribute” equally over all directions

• Total out-scattered radiance proportional to total radiance
  – integrate: $<L_c|1>$
Propagation Scheme
Propagation Scheme

- Base propagation same as LPV
- $\Delta L_c = \langle L_c | \Gamma_i \rangle \Gamma_i$
Propagation Scheme

- Extinction proportional to base propagation
- $\lambda \langle L_c | \Gamma_i \rangle \Gamma_i$
Propagation Scheme

- In-scattering proportional to total radiance in cell
- \( + \gamma_i \lambda' \langle L_c | 1 \rangle \Gamma_i \)
Propagation Scheme - Summary

• “base propagation” – extinction + in-scattering
  – base propagation: \( <L_c | \Gamma_i > \Gamma_i \)
  – extinction: \(-\lambda <L_c | \Gamma_i > \Gamma_i \)
  – in-scattering: \(+\gamma_i \lambda' <L_c | 1> \Gamma_i \)

• Gives (Equation 3 in Paper):

\[
(1 - b_{cc'})((1 - \lambda) <L_c | \Gamma_i > + \gamma_i \lambda' <L_c | 1>) \Gamma_i
\]
Overview – Implementation

- Introduction
  - Why & How
- Related Work
  - Single Scattering
  - Light Propagation Volumes
- Propagation Scheme
- Implementation
- Results & Conclusion
Implementation - Choices

- Each cell has 26 neighbors
  - nearest cells in grid
- Use 4-coefficient SHs (2 bands)
  - for radiances, transfer functions, ...
- Need to store & compute separate values for each RGB color channel
  - Each cell: 3x4 = 12 floating point values
- Use $32^3$ Grid
Implementation – Injection

• Each frame: start by generating an initial radiance distribution

• Inject
  – radiance from single scattering
    • identified by shadow map
  – radiance from reflective shadow maps
    • as in original LPV method
Implementation - Propagation

• With 4-coefficient SHs
  – Projection, $<L_c | \Gamma_i > \Gamma_i$, becomes a 4-dot product + element wise multiplication
  – Integration, $<L_c | 1>$, becomes a single multiplication

• So propagation (Equation 3 in paper) is a bunch of muls and adds:

$$(1-b_{cc})(1-\lambda)<L_c | \Gamma_j > + \gamma_j \lambda_s <L_c | 1> \Gamma_j$$
Implementation - Propagation

- With 4-coefficient SHs
  - Projection, $\langle L_c | \Gamma_i \rangle \Gamma_i$, becomes a 4-dot product + element wise multiplication
  - Integration, $\langle L_c | 1 \rangle$, becomes a single multiplication

- So propagation (Equation 3 in paper) is a bunch of muls and adds:

\[
(1-b_{cc'})((1-\lambda)\langle L_c | \Gamma_j \rangle + \gamma_j \lambda_s <L_c | 1>) \Gamma_j
\]
Implementation - Propagation

• One propagation iteration:
  – for each cell, compute contribution from each neighbor
  – separately for each color channel

• Perform a number of propagation iterations
  – $N = 8$ for $32^3$ seems sufficient
  – See original LPV paper
Implementation - Rendering

• Sample LPV for additional diffuse light
• Ray march LPV
  – currently not very clever, could be improved.
Implementation - Notes

• Tunable look
• Render different “effects” separately
  – single scattering
  – indirect illumination
  – multiple scattering in medium
• Inject from different sources separately
  – scattering
  – reflected radiance from RSMs
Implementation - Notes

- Everything computed each frame
  - from scratch

- Transfer functions only depend on lattice layout (grid)
  - while precomputed, these are completely independent from your scene.
Overview – Results & Conclusion

• Introduction
  – Why & How

• Related Work
  – Single Scattering
  – Light Propagation Volumes

• Propagation Scheme

• Implementation

• Results & Conclusion
Results

New Method
Results

Single Scattering Only
Results

LPV Only
Results

LPV and Single Scattering  New Method
Results - Performance

• Setup:
  – NVIDIA GTX 480, 1280x720
  – 256^2 RSM for injection
  – 2048^2 SM for shadows
  – 6x 256^2 additional shadowmaps for better blocking
  – rendered separately

• Details next slide

• Sponza, 32^3 LPV
  – ~30 FPS with full resolution ray marching
  – ~45 FPS at half resolution

• Sibenik, 32^3 LPV
  – ~35 FPS, ~50 FPS
## Results - Performance

<table>
<thead>
<tr>
<th>Sponza</th>
<th>$32^3$</th>
<th>$64^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total - Injection</strong></td>
<td>5.9 ms</td>
<td>5.2 ms</td>
</tr>
<tr>
<td>reflected light</td>
<td>0.87 ms</td>
<td>0.84 ms</td>
</tr>
<tr>
<td>blockers</td>
<td>4.5 ms</td>
<td>3.9 ms</td>
</tr>
<tr>
<td>single-scattered light</td>
<td>0.47 ms</td>
<td>0.59 ms</td>
</tr>
<tr>
<td><strong>Total - Propagation</strong></td>
<td>2.5 ms</td>
<td>17.0 ms</td>
</tr>
<tr>
<td>prepare blockers</td>
<td>0.40 ms</td>
<td>2.8 ms</td>
</tr>
<tr>
<td>propagate 8x</td>
<td>1.49 ms</td>
<td>12.3 ms</td>
</tr>
<tr>
<td>copy-back results</td>
<td>0.18 ms</td>
<td>1.24 ms</td>
</tr>
</tbody>
</table>
## Results - Performance

- Fairly independent of grid resolution
- High resolution grid may even help (fewer collisions during injection)

<table>
<thead>
<tr>
<th></th>
<th>Sponza 32^3</th>
<th>Sponza 64^3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sponza</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total - Injection</strong></td>
<td><strong>5.9 ms</strong></td>
<td><strong>5.2 ms</strong></td>
</tr>
<tr>
<td>reflected light</td>
<td>0.87 ms</td>
<td>0.84 ms</td>
</tr>
<tr>
<td>blockers</td>
<td>4.5 ms</td>
<td>3.9 ms</td>
</tr>
<tr>
<td>single-scattered light</td>
<td>0.47 ms</td>
<td>0.59 ms</td>
</tr>
<tr>
<td><strong>Total - Propagation</strong></td>
<td><strong>2.5 ms</strong></td>
<td><strong>17.0 ms</strong></td>
</tr>
<tr>
<td>prepare blockers</td>
<td>0.40 ms</td>
<td>2.8 ms</td>
</tr>
<tr>
<td>propagate 8x</td>
<td>1.49 ms</td>
<td>12.3 ms</td>
</tr>
<tr>
<td>copy-back results</td>
<td>0.18 ms</td>
<td>1.24 ms</td>
</tr>
</tbody>
</table>
### Results - Performance

- **Currently injecting a lot of blockers**
- **Could get away with fewer but better blockers...**

<table>
<thead>
<tr>
<th>Sponza</th>
<th>32^3</th>
<th>64^3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total - Injection</strong></td>
<td><strong>5.9 ms</strong></td>
<td><strong>5.2 ms</strong></td>
</tr>
<tr>
<td>reflected light</td>
<td>0.87 ms</td>
<td>0.84 ms</td>
</tr>
<tr>
<td>blockers</td>
<td><strong>4.5 ms</strong></td>
<td><strong>3.9 ms</strong></td>
</tr>
<tr>
<td>single-scattered light</td>
<td>0.47 ms</td>
<td>0.59 ms</td>
</tr>
<tr>
<td><strong>Total - Propagation</strong></td>
<td><strong>2.5 ms</strong></td>
<td><strong>17.0 ms</strong></td>
</tr>
<tr>
<td>prepare blockers</td>
<td>0.40 ms</td>
<td>2.8 ms</td>
</tr>
<tr>
<td>propagate 8x</td>
<td>1.49 ms</td>
<td>12.3 ms</td>
</tr>
<tr>
<td>copy-back results</td>
<td>0.18 ms</td>
<td>1.24 ms</td>
</tr>
</tbody>
</table>
## Results - Performance

- Obviously affected by grid resolution
- Probably should increase number of propagation iterations for 64^3

<table>
<thead>
<tr>
<th></th>
<th>Sponza</th>
<th>32^3</th>
<th>64^3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total - Injection</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reflected light</td>
<td>0.87 ms</td>
<td>0.84 ms</td>
<td></td>
</tr>
<tr>
<td>blockers</td>
<td>4.5 ms</td>
<td>3.9 ms</td>
<td></td>
</tr>
<tr>
<td>single-scattered light</td>
<td>0.47 ms</td>
<td>0.59 ms</td>
<td></td>
</tr>
<tr>
<td><strong>Total - Propagation</strong></td>
<td><strong>2.5 ms</strong></td>
<td><strong>17.0 ms</strong></td>
<td></td>
</tr>
<tr>
<td>prepare blockers</td>
<td>0.40 ms</td>
<td>2.8 ms</td>
<td></td>
</tr>
<tr>
<td>propagate 8x</td>
<td>1.49 ms</td>
<td>12.3 ms</td>
<td></td>
</tr>
<tr>
<td>copy-back results</td>
<td>0.18 ms</td>
<td>1.24 ms</td>
<td></td>
</tr>
</tbody>
</table>
Results - Limitations

- Low LPV resolution...
- Extreme initial distributions
  - “blocky” results
- Leakage with thin geometry
  - like original LPVs
- May be improved by using cascades as suggested in the original paper.
Results - Comparison

- Top: Offline Reference Image (approx 10h CPU)
  - full HDR, tonemapped, ...
  - physically based model

- Bottom: Rendering by our method (~ 35FPS)
  - tunable