ITERATIVE DEPTH WARING

Sungkil LEE*
Sungkyunkwan University (SKKU), South Korea

Younguk KIM

Elmar EISEMANN
Delft University of Technology, Netherlands
PHOTOGRAPHY & RECORDING ENCOURAGED
Depth Buffer

• **A fundamental basis for modern GPU rendering pipeline**
  • Many visibility-related operations
  • e.g., shadows, GI, occlusion culling, ambient occlusion, ...
  • Typically created in a full rendering pass

• **Research Goal: Early Access to Depth Buffer**
  • Depth-buffer prediction beneficial in many scenarios
Example: Occlusion Culling

- 6x faster (6.2 ms for 110M Tris., 94K objs., GTX1080)
Our Solution: Warping

• **Depth-Buffer Warping**
  • previous solutions mostly successful for color images
  • depth buffer needs to be precise/conservative

• **Challenge: real-time high-quality warping**
  • More precisely, *backward warping*
Forward vs. Backward Warping

- **Forward warping** splats source pixels $p$ (with motion vector) to destination pixels $q$.

\[
p = T^{-1}(q)
\]

- **Backward warping** gathers $p$ whose motions can displace to $q$.

\[
q = T(p)
\]
## Forward vs. Backward Warping in GPU

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Forward warping</strong></td>
<td><strong>GPU-unfriendly</strong> (splatting or atomics)</td>
</tr>
<tr>
<td>• knows destination pixels to write</td>
<td>• holes may appear</td>
</tr>
<tr>
<td><strong>Backward warping</strong></td>
<td><strong>Need non-trivial search</strong> when source pixels are unknown (arbitrary motions)</td>
</tr>
<tr>
<td>• GPU-friendly (only texture lookups), and usually faster</td>
<td></td>
</tr>
<tr>
<td>• no holes (when source exists)</td>
<td></td>
</tr>
<tr>
<td>• Preferred in general</td>
<td></td>
</tr>
</tbody>
</table>
Our Contributions

• Our solution builds upon the previous image warping
  • Iterative Image Warping [Bowles et al. 2011]
  • but, for warping of depth buffer

• Our solution improves on:
  • purely backward-only warping
  • efficient bounding of backward search area
  • multi-layer warping for depth hole filling
Related Work and Background
Temporal Coherence

- Temporal cache of shading values [Nehab et al. 2007]
  - Reverse reprojection can reuse shading
  - still requires geometry pass
Multi-Source Image Warping

- Post-rendering 3D Warping [Mark et al. 1997]
- Forward mapping of per-pixel primitives for reference frames
Multi-Source Image Warping

- Bidirectional reprojection [Yang et al. 2011]
  - Disocclusion holes filled with bidirectional B-frames
  - Delay in frame time
Single-Source Image Warping

- Grid-based forward warping
  - Temporal upsampling, stereo-view synthesis [Didyk et al., 2010a, b]
  - Coarse/adaptive grid subdivision

[Didyk 2010a] [Didyk 2010b]
Single-Source Image Warping

- Fast backward warping [Andreev 2010]
  - Mostly for background
  - Simple inpainting: replication from neighboring patches
  - Suitable for frame rate upsampling
Single-Source Image Warping

- **Fixed-point iteration** [Bowles et al., 2011]
  - Backward warping formulated as fixed-point iteration
  - Hybrid warping: *forward pre-warping* + backward warping
  - Costly tessellation required
Background: Fixed-Point Iteration (FPI)

• A fixed point $\tilde{x}$ of function $g$ satisfies: $\tilde{x} = g(\tilde{x})$.
• FPI $x_{n+1} \leftarrow g(x_n)$ converges to $\tilde{x} = g(\tilde{x})$
• when $g$ is differentiable around $\tilde{x}$ and $|g'(\tilde{x})| < 1$. 
Our Approach
Inputs

- Depth buffer $Z$ and motion buffer $V$
- from the previous frame or the known view
- Readily available in deferred rendering pipeline

Depth buffer $Z$  
Motion vector $V$
Outputs

• **Estimated depth buffer \( \hat{z} \) at novel views**
  • Hole filling used optionally, depending on applications

Conservative without hole filling

With multi-layer hole filling
Backward Warping with FPI

• Given a destination pixel $p$, FPI on source pixels $q$:

$$ q_{i+1} \leftarrow w(q_i) := p - v(q_i) $$

• Iteration stops at a fixed point $\tilde{q} = w(\tilde{q})$. In other words, $\tilde{q}$ displaces to $p$:

$$ \tilde{q} + v(\tilde{q}) = p $$
Simple Example

- Choose the closest among multiple solutions
- For a smooth surface, 2—3 iteration is enough

No motion!

$p$: destination in novel view

Search seeds

Solution
Problems

• **Seed $q_0$ is crucial.**
  • A simple $q_0 = p$ may fail at boundaries between different motions

• **Convergence can be guaranteed:**
  • with 2×2 Jacobian matrix of $w$ [Bowles 2012]
  • but, requires costly fine tessellation and pre-warping
Our solution

• Stratified sampling in compact local search regions $\Omega$
Our solution

• **Stratified sampling in compact local search regions** $\Omega$
  • May not be precise for a complex scene
  • Practically, higher sampling density leads to convergence.

• **Motion bound query: use** min-max mipmap
  • min/max of (x,y) motion can be encoded in 4-channel texture
    • `vec4( xmotion_min, xmotion_max, ymotion_min, ymotion_max )`
  • Refer to the paper for the pseudocode
Results: Single-Layer Warping

• Set **background depth** for depth holes (disocclusions)
  • Conservative estimation
Multi-layer Warping for Depth Hole Filling

• We may use conservative estimation (e.g., occlusion culling)
  • But, we may need to fill the holes for a higher accuracy

• Our solution: **multi-layer inpainting**
  • Repeat search for hidden layers
Generating Hidden Layers

- Depth peeling [Everitt 2001]
Generating **Effective** Hidden Layers

- Depth peeling often requires too many layers
  - Effective reduction of layers: peeling of only visible objects
  - We generalize lens umbra culling [Lee 2010] for motions

![Diagram showing umbra and depth peeling threshold between known and novel views](image-url)
Depth Peeling with Umbra Culling

• Reduces ~2 layers for similar warping quality
Results: Multi-Layer Warping

• Useful for multiview rendering (e.g., soft shadows)
Experimental Analysis
Test Configurations

• **Experimental configurations**
  • NVIDIA GTX 1080, Full-HD (1920×1080), OpenGL 4.5
  • Four camera-animated scenes (8.5M—110M faces, 17—94K objects)
Search Seed Samples (N) and Layers (L)

- More samples/layers give better results (PSNR/SSIM)
  - $L=1$ (single-layer): good, except for depth holes
  - $L=4$ (multi-layer): try to saturate $N \geq 4$ (37 dB, 0.997)
  - Practical speed/performance tradeoff: $N=[8,16]$
Quality Comparison with Forward Warping

- Two common forward warping techniques
  - VFW (Vertex-based FW): vertex-to-quad splatting
  - AFW (Atomics-based FW): direct writes with compute shader

- Results (averaged on all the animation sequences)
  - Ours: similar SSIM, but much higher in PSNR (better pixel accuracy)

<table>
<thead>
<tr>
<th></th>
<th>Ours (N=16)</th>
<th>Vertex FW</th>
<th>Atomics FW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>SSIM</td>
<td>PSNR</td>
</tr>
<tr>
<td>L=1 (single-layer)</td>
<td>28.1 dB</td>
<td>0.977</td>
<td>24.0 dB</td>
</tr>
<tr>
<td>L=4 (multi-layer)</td>
<td>34.3 dB</td>
<td>0.997</td>
<td>28.1 dB</td>
</tr>
</tbody>
</table>
Number Iteration (T)

- $T=2$—3 suffices for moderate motions, but requires more iteration for rapid motions
## Warping Performance

<table>
<thead>
<tr>
<th>Ours (N=16)</th>
<th>Vertex FW</th>
<th>Atomics FW</th>
</tr>
</thead>
<tbody>
<tr>
<td>L=1</td>
<td>0.68 ms</td>
<td>2.2 ms</td>
</tr>
<tr>
<td>L=4</td>
<td>0.93 ms</td>
<td>6.2 ms</td>
</tr>
</tbody>
</table>

- **Ours: less than 1 ms**
  - additional 0.1 ms, when needing motion generation.

- **Comparison with FW**
  - Much faster than classical VFW
  - Slightly slower than newer AFW (0.38—0.45 ms)
Application: Occlusion Culling
Hierarchical Occlusion Mapping Revisited

Object-space bound

Max-depth query (N-buffers) < Min-depth query (bounding corners)

Occlusion culled
Warping-based Occlusion Culling (WOC)

• Use the estimated depth buffer for HOM
  • No need to distinguish occluder/occludee, unlike the classical ones
  • L=1 sacrifices the culling for depth holes
  • But, its negligible overhead eventually improves performance
WOC Performance

• Balloons (33M Tris., 35,200 objects)
WOC Performance

• Cityblock (110M Tris., 94,275 objects)
Application: Soft-Shadow Mapping
Speed-up of 1—30X

• More efficient for: complex scenes, many lights, smaller maps
Limitations and Conclusion
Limitations

- Rapid motions and flipping triangles
  - e.g., faster than ~200 pixels per frame
  - Practical remedy: dilation around neighbors
Limitations

• Sub-pixel accuracy
  • From the rasterized pixel precision
  • Rapid motion vectors amplify the errors

• Image periphery
  • Need to extend the image FOV during the capture
Conclusion

• Scalable high-quality depth warping, based on
  • tight bounding technique for pure backward warping
  • multi-layer depth hole filling

• Suitable for real-time GPU rendering
  • Scalable warping-based occlusion culling (WOC)
  • Potentially useful many multi-view rendering scenarios
    • Depth-of-field, anti-aliasing, intra-frame reprojection for VR
Thank you for attention!