Frustum Volume Caching for Accelerated NeRF Rendering

Michael Steiner¹, Thomas Köhler¹, Lukas Radl¹, Markus Steinberger¹,²

¹ Graz University of Technology, Austria
² Huawei Technologies, Austria
Introduction to NeRFs
Neural Radiance Fields (NeRFs)

- Inverse rendering: Learning a 3D scene from 2D images
- NeRF [Mildenhall et al. 2020]
  - Differentiable volumetric rendering
  - Outgoing radiance $\Theta : (x, d) \mapsto (c, \sigma)$
  - Large MLPs (~10h training, <0.1 FPS)

Source: Mildenhall et al., “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. ECCV (2020)
Background - NeRF Models

- **Unbounded, anti-aliased NeRF: Mip-NeRF 360** [Barron et al., 2022]
  - Scene contraction for unbounded scenes
  - Distortion loss: encourage NeRF to form proper surfaces
- **Instant-NGP** [Müller et al. 2022]
  - Multi-resolution hash encoding
  - Occupancy grids for empty space skipping
  - Fast training and rendering (5-10 min. training, ~10 FPS)
Background - NeRF Network

- Expensive base extracts view-independent latent code
- Smaller head outputs view-dependent color
Motivation - Temporal Coherence
Background & Related Work
Related Work - Accelerating NeRFs

- “Baking” NeRFs into a render-friendly format:
  - SNeRG: Sparse voxel grid [Hedman et al., 2021]
  - Mesh-based: e.g. MobileNeRF [Chen et al., 2023]

- Fast-NeRF [Garbin et al., 2021]
  - Split NeRF into two completely separate networks
  - Cache in world space
Related Work – Volumetric Temporal Coherence Methods

- Image-based [Mueller, 1999]
- 3D Irradiance Probes [Greger, 1998]
- Neural Radiance Caching [Müller et al., 2021]
- View-aligned Cache:
  - Frustum Voxel (Froxel) Grid [Wronski 2014, Hillaire 2015]
Caching NeRFs

Idea & Challenges
Volumetric Caching of NeRF Samples
Challenges

- Per-sample cache hit/miss detection
- Reasonable Cache Size
  - Small Latent Codes
  - Sparse Datastructure
- Fast Cache Initialization
- Critical: Fast cache lookups and color re-evaluation
- Interpolating samples from Cache
Interpolating NeRF Samples
Interpolation

- Cache consists of three separate 3D Froxel Grids:
  - $Z_1$ ... Latent codes
  - $Z_\sigma$ ... Density
  - $Z_o$ ... Binary occupancy information

- Unoccupied cells initialized to zero

- Transform 3D sample point $x$ into froxel space point $z$

$$\hat{\sigma} = \text{trilerp}(z, Z_\sigma), \quad \hat{I} = \frac{\text{trilerp}(z, Z_1)}{\text{trilerp}(z, Z_o)}$$

- **Problem:** Linearly interpolating neural network output!

Interpolating NeRF Samples
Naïve Latent Code Interpolation
Interpolating NeRF Samples

Learning Spatial Linearity

- Interpolate during training with random stepping offset $\Delta z \in [0, 1]$

\[
\Delta z \cdot \delta_i \quad x_{(i,1)} = r(t_i + (1 - \Delta z)\delta_i) \\
\quad \delta_i \quad x_{(i,0)} = r(t_i - (1 - \Delta z)\delta_i) \\
\quad t_{i-1} \quad \approx x_{(i-1,1)}
\]

- If any sample is outside the occupancy grid → set value to zero

\[
\hat{\sigma} = \text{lerp}(\Delta z, o_0\sigma_{x_0}, o_1\sigma_{x_1}), \quad \hat{l} = \frac{\text{lerp}(\Delta z, o_0 l_{x_0}, o_1 l_{x_1})}{\text{lerp}(\Delta z, o_0, o_1)}
\]
View-Dependent Cone Encoding (Idea)

- Latent codes are completely view-independent
- We only allow small view-changes
- Advantages:
  - Latent codes need to be less informative
  - Head network can be smaller
View-Dependent Cone Encoding

- Provide difference in encoded view-direction to final MLP
- During optimization:
  - Generate 4 randomly shifted directions
  - Average losses → learn relationship between view-directions
- Shift computational load to new \textbf{neck} network
A Model for Efficient Rendering From Cache

- Instant-NGP + View-Dependent Cone Encoding
- Scene contraction + Distortion Loss: reduce sample count
NeRF Frustum Volume Caching & Reprojection
Cache Datastructure & Initialization

- Sparse froxel grid with fixed-size bricks (e.g. 8x8x8)
- Neighboring in $B \neq$ neighboring in $D$

- Initialization:
  - Brick-wise (max. utilization)
  - Pad bricks (optional)
Reprojection & Sampling

- Per-sample: Occupancy grid not set → Early reject
- Cache-miss: only if sample is outside known range $K$
- Cache-hit: only if inside $K$, $B_O$, and $D_O$ + high enough opacity

\[
\begin{align*}
\text{Indices inside } B_I \\
\text{point into } D
\end{align*}
\]
Implementation Details

- Training framework: Pytorch, based on Nerfacc [Li et al., 2023]
- Real-time Viewer & Offline Renderer: C++/CUDA
  - tiny-cuda-nn: Hash-grid encoding & MLPs
  - Fused head network
Results

Quantitative and Qualitative
Dataset

- Mip-NeRF 360 [Barron et al., 2021]
  - Challenging indoor and outdoor scenes
  - 360° with central object

Models

- iNGP (big)
- iNGP *Ours*
  - Scene contraction, distortion loss, interpolation training
- *Ours*
  - Cone enc.: $\theta_{\text{max}} = 25°$
  - Smaller latent codes
  - Smaller head network
- *Ours* (huge)
  - Larger hash-grid
Model Quality

- Competitive quality with reduced sample counts (~12 spp)
  - Despite smaller latent codes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mip-NeRF 360 Indoor</th>
<th></th>
<th>Mip-NeRF 360 Outdoor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>PSNR↑</td>
<td>SSIM↑</td>
<td>LPIPS↓</td>
<td>mean(N_s)↓</td>
</tr>
<tr>
<td>Plenoxels‡</td>
<td>24.84</td>
<td>0.765</td>
<td>0.366</td>
<td>-</td>
</tr>
<tr>
<td>Mip-NeRF 360‡</td>
<td>31.57</td>
<td>0.914</td>
<td>0.182</td>
<td>-</td>
</tr>
<tr>
<td>3DGS‡</td>
<td>30.41</td>
<td>0.920</td>
<td>0.190</td>
<td>-</td>
</tr>
<tr>
<td>iINGP (big)†</td>
<td>29.44</td>
<td>0.866</td>
<td>0.257</td>
<td>35.02</td>
</tr>
<tr>
<td>iINGP Ours</td>
<td>30.32</td>
<td>0.889</td>
<td>0.219</td>
<td>12.94</td>
</tr>
<tr>
<td>Ours</td>
<td>30.14</td>
<td>0.891</td>
<td>0.220</td>
<td>12.92</td>
</tr>
<tr>
<td>Ours (huge)</td>
<td>30.58</td>
<td>0.901</td>
<td>0.203</td>
<td>12.53</td>
</tr>
</tbody>
</table>
## Model Quality

- Competitive quality with reduced sample counts (~12 spp)
  - Despite smaller latent codes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mip-NeRF 360 Indoor</th>
<th>Mip-NeRF 360 Outdoor</th>
<th>iNGP (big)†</th>
<th>iNGP Ours</th>
<th>Ours</th>
<th>Ours (huge)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>PSNR↑</td>
<td>SSIM↑</td>
<td>LPIPS↓</td>
<td>mean(Nₛ)↓</td>
<td>PSNR↑</td>
<td>SSIM↑</td>
</tr>
<tr>
<td>Plenoxels‡</td>
<td>24.84</td>
<td>0.765</td>
<td>0.366</td>
<td>-</td>
<td>21.69</td>
<td>0.513</td>
</tr>
<tr>
<td>Mip-NeRF 360‡</td>
<td>31.57</td>
<td>0.914</td>
<td>0.182</td>
<td>-</td>
<td>24.18</td>
<td>0.653</td>
</tr>
<tr>
<td>3DGS‡</td>
<td>30.41</td>
<td>0.920</td>
<td>0.190</td>
<td>-</td>
<td>30.32</td>
<td>0.889</td>
</tr>
</tbody>
</table>

Michael Steiner
“Frustum Volume Caching for Accelerated NeRF Rendering”, HPG’24
Visualization of our Evaluation Setup

Cache Render View is Static
Cache Initialization View moves around
Caching & Reprojection - Performance

- Much faster than *iNGP (big)* due to low sample count

<table>
<thead>
<tr>
<th>Type Method</th>
<th>No Cache</th>
<th>( t = \frac{\Delta t_{\text{min}}}{2} ) CHR ~97%</th>
<th>( \theta = 5^\circ ) CHR ~83%</th>
<th>( \theta = 10^\circ ) CHR ~72%</th>
<th>( \theta = 15^\circ ) CHR ~63%</th>
<th>Avg. Cache Size (GiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Ours</em></td>
<td>48.82</td>
<td>25.68 (1.90×)</td>
<td>28.38 (1.72×)</td>
<td>30.64 (1.59×)</td>
<td>32.63 (1.50×)</td>
<td>3.41</td>
</tr>
<tr>
<td><em>iNGP (big)</em>‡</td>
<td>124.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><em>iNGP Ours</em></td>
<td>46.79</td>
<td>43.32 (1.08×)</td>
<td>42.69 (1.10×)</td>
<td>42.35 (1.10×)</td>
<td>42.46 (1.10×)</td>
<td>5.69</td>
</tr>
<tr>
<td><em>iNGP Ours N_l = 8</em></td>
<td>49.44</td>
<td>42.46 (1.16×)</td>
<td>42.50 (1.16×)</td>
<td>42.56 (1.16×)</td>
<td>43.01 (1.15×)</td>
<td>3.58</td>
</tr>
</tbody>
</table>

Render timings in ms
### Caching & Reprojection - Performance

- Large speedups when rendering from cache
- \textit{iNGP} unsuited for caching; almost no speedup

<table>
<thead>
<tr>
<th>Type Method</th>
<th>No Cache</th>
<th>$t = -\frac{\Delta t_{\text{min}}}{2}$</th>
<th>$\theta = 5^\circ$</th>
<th>$\theta = 10^\circ$</th>
<th>$\theta = 15^\circ$</th>
<th>Avg. Cache Size (GiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CHR $\sim$97%</td>
<td>CHR $\sim$83%</td>
<td>CHR $\sim$72%</td>
<td>CHR $\sim$63%</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>48.82</td>
<td>25.68 (1.90×)</td>
<td>28.38 (1.72×)</td>
<td>30.64 (1.59×)</td>
<td>32.63 (1.50×)</td>
<td>3.41</td>
</tr>
<tr>
<td>iNGP (big)†</td>
<td>124.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>\textit{iNGP Ours}</td>
<td>46.79</td>
<td>43.32 (1.08×)</td>
<td>42.69 (1.10×)</td>
<td>42.35 (1.10×)</td>
<td>42.46 (1.10×)</td>
<td>5.69</td>
</tr>
<tr>
<td>\textit{iNGP Ours $N_l = 8$}</td>
<td>49.44</td>
<td>42.46 (1.16×)</td>
<td>42.50 (1.16×)</td>
<td>42.56 (1.16×)</td>
<td>43.01 (1.15×)</td>
<td>3.58</td>
</tr>
</tbody>
</table>

Render timings in ms
Caching & Reprojection - Performance

- Cache Size smaller due to smaller latent codes

<table>
<thead>
<tr>
<th>Type Method</th>
<th>No Cache</th>
<th>$t = \frac{\Delta t_{\text{min}}}{2}$ CHR $\sim$97%</th>
<th>$\theta = 5^\circ$ CHR $\sim$83%</th>
<th>$\theta = 10^\circ$ CHR $\sim$72%</th>
<th>$\theta = 15^\circ$ CHR $\sim$63%</th>
<th>Avg. Cache Size (GiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>48.82</td>
<td>25.68 (1.90×)</td>
<td>28.38 (1.72×)</td>
<td>30.64 (1.59×)</td>
<td>32.63 (1.50×)</td>
<td>3.41</td>
</tr>
<tr>
<td>iNGP (big)$^\dagger$</td>
<td>124.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>iNGP Ours</td>
<td>46.79</td>
<td>43.32 (1.08×)</td>
<td>42.69 (1.10×)</td>
<td>42.35 (1.10×)</td>
<td>42.46 (1.10×)</td>
<td>5.69</td>
</tr>
<tr>
<td>iNGP Ours $N_l = 8$</td>
<td>49.44</td>
<td>42.46 (1.16×)</td>
<td>42.50 (1.16×)</td>
<td>42.56 (1.16×)</td>
<td>43.01 (1.15×)</td>
<td>3.58</td>
</tr>
</tbody>
</table>

Render timings in ms
Caching & Reprojection - Quality

- Our modifications enable high quality from-cache rendering
  - *Ours* can faithfully render from cache even for small latent codes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mip-NeRF 360 Indoor</th>
<th>Mip-NeRF 360 Outdoor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Cache</td>
<td>5°</td>
</tr>
<tr>
<td><strong>Method</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Ours</em></td>
<td>30.12</td>
<td>30.25</td>
</tr>
<tr>
<td><em>Ours</em> w/o z-linearity</td>
<td>30.37</td>
<td>24.75</td>
</tr>
<tr>
<td><em>Ours</em> 5°</td>
<td>30.35</td>
<td>30.29</td>
</tr>
<tr>
<td><em>Ours</em> Nₜ = 4</td>
<td>30.03</td>
<td>30.16</td>
</tr>
<tr>
<td>iNGP Ours w/o z-linearity</td>
<td>30.56</td>
<td>29.84</td>
</tr>
<tr>
<td>iNGP Ours Nₜ = 8</td>
<td>30.15</td>
<td>29.70</td>
</tr>
</tbody>
</table>
Caching & Reprojection - Quality

- Our modifications enable high quality from-cache rendering
  - *Ours* can faithfully render from cache even for small latent codes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mip-NeRF 360 Indoor</th>
<th></th>
<th>Mip-NeRF 360 Outdoor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Cache</td>
<td>5°</td>
<td>10°</td>
<td>15°</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>30.12</td>
<td>30.25</td>
<td>30.18</td>
<td>30.05</td>
</tr>
<tr>
<td><strong>Ours w/o z-linearity</strong></td>
<td>30.27</td>
<td>24.52</td>
<td>24.65</td>
<td>24.73</td>
</tr>
<tr>
<td><strong>Ours 5°</strong></td>
<td>30.35</td>
<td>30.29</td>
<td>29.60</td>
<td>28.34</td>
</tr>
<tr>
<td><strong>Ours N_1 = 4</strong></td>
<td>30.03</td>
<td>30.16</td>
<td>30.08</td>
<td>29.95</td>
</tr>
<tr>
<td><strong>iNGP Ours w/o z-linearity</strong></td>
<td>30.56</td>
<td>29.84</td>
<td>29.83</td>
<td>29.81</td>
</tr>
<tr>
<td><strong>iNGP Ours N_1 = 8</strong></td>
<td>30.15</td>
<td>29.70</td>
<td>29.68</td>
<td>29.64</td>
</tr>
</tbody>
</table>
Caching & Reprojection - Quality

- Our modifications enable high quality from-cache rendering
  - *Ours* can faithfully render from cache even for small latent codes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mip-NeRF 360 Indoor</th>
<th></th>
<th></th>
<th>Mip-NeRF 360 Outdoor</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Cache</td>
<td>Rotation</td>
<td>Translation</td>
<td>No Cache</td>
<td>Rotation</td>
<td>Translation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5°</td>
<td>10°</td>
<td>15°</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>Ours</td>
<td>30.12</td>
<td>30.25</td>
<td>30.18</td>
<td>30.05</td>
<td>30.29</td>
<td>30.14</td>
</tr>
<tr>
<td>Ours w/o z-linearity</td>
<td>30.27</td>
<td>24.52</td>
<td>24.65</td>
<td>24.73</td>
<td>25.73</td>
<td>23.53</td>
</tr>
<tr>
<td>Ours 5°</td>
<td>30.35</td>
<td>30.29</td>
<td>29.60</td>
<td>28.34</td>
<td>30.49</td>
<td>30.33</td>
</tr>
<tr>
<td>Ours 45°</td>
<td>30.02</td>
<td>30.15</td>
<td>30.09</td>
<td>29.99</td>
<td>30.19</td>
<td>30.04</td>
</tr>
<tr>
<td>Ours N₁ = 4</td>
<td>30.03</td>
<td>30.16</td>
<td>30.08</td>
<td>29.95</td>
<td>30.20</td>
<td>30.07</td>
</tr>
<tr>
<td>iNGP Ours</td>
<td>30.30</td>
<td>30.43</td>
<td>30.37</td>
<td>30.28</td>
<td>30.47</td>
<td>30.31</td>
</tr>
<tr>
<td>iNGP Ours w/o z-linearity</td>
<td>30.56</td>
<td>29.84</td>
<td>29.83</td>
<td>29.81</td>
<td>30.10</td>
<td>29.67</td>
</tr>
<tr>
<td>iNGP Ours N₁ = 8</td>
<td>30.15</td>
<td>29.70</td>
<td>29.68</td>
<td>29.64</td>
<td>29.89</td>
<td>29.54</td>
</tr>
</tbody>
</table>

Michael Steiner
“Frustum Volume Caching for Accelerated NeRF Rendering”, HPG’24
Caching & Reprojection - Quality

- Our modifications enable high quality from-cache rendering
  - *Ours* can faithfully render from cache even for small latent codes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mip-NeRF 360 Indoor</th>
<th></th>
<th>Mip-NeRF 360 Outdoor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rotation</td>
<td>Translation</td>
<td>Rotation</td>
<td>Translation</td>
</tr>
<tr>
<td></td>
<td>No Cache</td>
<td>5°</td>
<td>10°</td>
<td>15°</td>
</tr>
<tr>
<td>Ours</td>
<td>30.12</td>
<td>30.25</td>
<td>30.18</td>
<td>30.05</td>
</tr>
<tr>
<td>Ours w/o z-linearity</td>
<td>30.27</td>
<td>24.52</td>
<td>24.65</td>
<td>24.73</td>
</tr>
<tr>
<td>Ours 5°</td>
<td>30.35</td>
<td>30.29</td>
<td>29.60</td>
<td>28.34</td>
</tr>
<tr>
<td>Ours N₁ = 4</td>
<td>30.03</td>
<td>30.16</td>
<td>30.08</td>
<td>29.95</td>
</tr>
<tr>
<td>iNGP Ours w/o z-linearity</td>
<td>30.56</td>
<td>29.84</td>
<td>29.83</td>
<td>29.81</td>
</tr>
<tr>
<td>iNGP Ours N₁ = 8</td>
<td>30.15</td>
<td>29.70</td>
<td>29.68</td>
<td>29.64</td>
</tr>
</tbody>
</table>
Results

Video Sequence – Setup

- 300 frame video
- Pre-defined camera path
- Ours (huge)
- Supersampled volumetric motion blur & depth of field
Without Caching

With Caching
(Speedup of 1.82x)
Video Sequence - Results

- Cache init. times included!
- Automatic Cache Rebuild
- Spikes for decreasing Cache-Hit-Rate
Conclusion, Limitations & Future Work
Limitations

- Large cache sizes
  - Not suitable for lower-end graphics devices
- Cache initialization
  - Latency hiding required for real-time rendering
- Still slower than explicit methods (e.g. meshes or 3DGS)

Future Work

- Reducing the memory footprint
  - Compression
- VR Application
Conclusion

- Accelerate NeRFs through caching and reprojection
  - View-aligned, fully volumetric cache
  - Re-evaluation of view-dependent effects
- High-quality, fast offline & real-time rendering
Thank you for your attention!

Public Source Code

Code: https://github.com/steimich96/FrustumVolumeCaching

Project-Page: https://steimich96.github.io/FrustumVolumeCaching