Towards Practical Applications of NeRF

Songyou Peng
Who Am I?

• PhD Student
  • Marc Pollefeys
  • Andreas Geiger

• Internships during PhD
  • 2021: Michael Zollhoefer
  • Now: Tom Funkhouser

• Graduate next summer

pengsongyou.github.io
My PhD Topics: Neural Scene Representations for 3D reconstruction, novel view synthesis, and SLAM

Convolutional Occupancy Networks
ECCV 2020 (Spotlight)

KiloNeRF
ICCV 2021

UNISURF
ICCV 2021 (Oral)

Shape As Points
NeurIPS 2021 (Oral)

NICE-SLAM
CVPR 2022

Ours

MonoSDF
arXiv 2022
NeRF is awesome!

Some problems still exist...

😊 Slow rendering speed
😊 KiloNeRF
😊 UNISURF + MonoSDF
😊 NICE-SLAM

😢 Slow rendering speed
😢 Poor underlying geometry
😢 Camera poses needed

KiloNeRF
Speeding up NeRF with Thousands of Tiny MLPs
Key Idea

- Partition a scene into a $16^3$ uniform grid
- Each grid cell is represented by a tiny MLP

87x reduction in FLOPs!

* FLOP: floating points operations
KiloNeRF

Training:

1. Distill a trained NeRF model into our KiloNeRF model
   - Randomly sampled points, their predicted alpha & color values should match!

2. Finetune the thousand MLPs on training images
KiloNeRF

Training:
1. Distill a trained NeRF model into our KiloNeRF model
   • Randomly sampled points, their predicted alpha & color values should match!
2. Finetune the thousand MLPs on training images

Inference:
1. Empty Space Skipping (ESS) with a pre-computed $256^3$ occupancy grid
2. Early Ray Termination (ERT): when transmittance $< \varepsilon$, stop!
3. Evaluate tiny MLPs in parallel
<table>
<thead>
<tr>
<th>Method</th>
<th>Render time ↓</th>
<th>Speedup ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF</td>
<td>56185 ms</td>
<td>–</td>
</tr>
<tr>
<td>NeRF + ESS + ERT</td>
<td>788 ms</td>
<td>71</td>
</tr>
<tr>
<td>KiloNeRF</td>
<td>22 ms</td>
<td>2554</td>
</tr>
</tbody>
</table>

* Tested with NVIDIA GTX 1080 Ti
Results
NeRF 800x800

56 s

KiloNeRF 800x800

0.02 s (50 fps)
runs now at 50 fps on a GTX 1080 Ti

https://github.com/creiser/kilonerf
## Comparison to Concurrent Works

<table>
<thead>
<tr>
<th>Type</th>
<th>Neural</th>
<th>Tabulation-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>KiloNeRF</td>
<td>PlenOctree</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNeRG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FastNeRF</td>
</tr>
<tr>
<td>GPU Memory</td>
<td>&lt; 100 MB</td>
<td>1930 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3442 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7830 MB</td>
</tr>
</tbody>
</table>

⇒ KiloNeRF has a larger potential for large-scale NVS!

BlockNeRF applied our idea for city-level NVS 😊
Take-home Message

• Speed up NeRF significantly (~2000x) without loss of quality
• A memory more friendly representation!

Limitations

• Only work on bounded scenes
• Expensive training time
UNISURF
Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction
Motivation

The underlying geometry of NeRF (volume rendering) is poor

Motivation

Surface rendering methods produce great geometry, but require object masks.

NeRF Rendering  NeRF Geometry  IDR [1] Geometry

Can we obtain accurate geometry without masks?

NeRF

IDR

UNISURF

UNISURF
Unify radiance fields and implicit surface model 😊
UNISURF

Early Stage: Volume rendering, but reformulate density to occupancy

NeRF rendering:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i(\mathbf{x}_i) \prod_{j<i} (1 - \alpha_j(\mathbf{x}_j)) \; c(\mathbf{x}_i, \mathbf{d}) \quad \alpha_i(\mathbf{x}) = 1 - \exp\left(-\sigma(\mathbf{x}) \delta_i\right)$$

Assuming a solid object, the alpha is the continuous occupancy field

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} o(\mathbf{x}_i) \prod_{j<i} (1 - o(\mathbf{x}_j)) \; c(\mathbf{x}_i, \mathbf{d})$$

1 for the first occupied sample
0 for all other samples

Points near to the surface have larger influence to the predicted color
UNISURF

Later Stage: Find surface points, decrease the range of volume rendering

a) Find the surface point

b) Define the new interval

c) Volume rendering
Loss Functions

a) Image reconstruction loss

\[ \mathcal{L}_{\text{rec}} = \sum_{r \in \mathcal{R}} \| \hat{C}_v(r) - C(r) \|_1 \]

b) Surface smoothness regularization

\[ \mathcal{L}_{\text{reg}} = \sum_{x_s \in \mathcal{S}} \| n(x_s) - n(x_s + \epsilon) \|_2 \]
Results
BlendedMVS
Take-home Message

• Volume rendering and implicit surfaces can be unified!
• Accurate reconstruction without the need of masks
• Many cocurrent & follow-up works: NeuS, VolSDF, NeuralWarp, GeoNeuS...

Limitations

• Hard to reconstruct texture-less regions
• Still limited to small object-centric scenes
• Won’t work given only sparse views
Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction

Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, Andreas Geiger
arXiv 2022
VolSDF / NeuS / UNISURF

Only supervision: multi-view RGB images
MonoSDF

\[
\mathcal{L}_{\text{normal}} = \sum_{r \in \mathcal{R}} \| \hat{\mathbf{N}}(r) - \tilde{\mathbf{N}}(r) \|_1 + \| 1 - \hat{\mathbf{N}}(r)^\top \tilde{\mathbf{N}}(r) \|_1
\]

\[
\mathcal{L}_{\text{depth}} = \sum_{r \in \mathcal{R}} \| (w \hat{\mathbf{D}}(r) + q) - \tilde{\mathbf{D}}(r) \|^2
\]
Results

Large-scale Indoor Scenes
Results

DTU with 3 Input Views
TSDF Fusion w/ Scaled Depths

VolSDF

[NeurIPS’21]

COLMAP

[ECCV’16]
Take-home Message

- Easy-to-obtain monocular cues are important!
- Also help converge faster and better!

Limitations

- Depends on the quality of the monocular cues
What is missing?

KiloNeRF

UNISURF

MonoSDF

runs now at 50 fps on a GTX 1080 Ti
NICE-SLAM
Neural Implicit Scalable Encoding for SLAM

CVPR 2022

Zihan Zhu*  Songyou Peng*  Viktor Larsson  Weiwei Xu  Hujun Bao
Zhaopeng Cui  Martin R. Oswald  Marc Pollefeys

* Equal Contributions
RGB-D Sequences

40x Speed
iMAP
[Sucar et al., ICCV'21]

First neural implicit-based **online** SLAM system
iMAP
[Sucar et al., ICCV’21]

A single MLP

- Fail when scaling up to larger scenes
- Global update → Catastrophic forgetting
- Slow convergence
NICE-SLAM

Applicable to large-scale scenes
Local update $\rightarrow$ No forgetting problem
Fast convergence

Feature grids + tiny MLPs

Predicted Poses
GT Poses
Pipeline

Mapping

Input Depth

Input RGB

Ray -> Point Sampler

Camera Pose

Hierarchical Feature Grid

Tri-linear Interpolation

Coarse-level Occupancy

\( \phi_p^0 \)

Coarse Level

Mid Level

Fine Level Occupancy

\( \phi_p^2 \)

\( \phi_p^1 \)

Color

\( \phi_p^0(p) \)

Hierarchical Feature Grid

Tri-linear Interpolation

\( \phi_p^0(p) \)

\( \phi_p^1(p) \)

\( \phi_p^2(p) \)

Ray -> Point Sampler

Camera Pose

Hierarchical Feature Grid

Tri-linear Interpolation

\( \phi_p^0(p) \)

\( \phi_p^1(p) \)

\( \phi_p^2(p) \)
Results
iMAP* (our re-implementation of iMAP)

NICE-SLAM

4x Speed

Predicted Poses
GT Poses
iMAP* (our re-implementation of iMAP) 10x Speed NICE-SLAM
Take-home Message

• A NICE online implicit SLAM system for indoor scenes
• Hierarchical feature grids + a tiny MLP seems to be a trend!
  • Instant-NGP [TOG]

Limitations

• Requires depths as input
• Only bounded scenes
• Still not real-time
Final Remarks

• NeRF has been sped up significantly for both rendering and optimization

• NeRF-based multi-view surface reconstruction still has rooms to improve

• A completely COLMAP-free NeRF pipeline?

• What is THE representation?
Thanks!

KiloNeRF  UNISURF  NICE-SLAM  MonoSDF