Towards Efficient Neural Scene Graphs by Learning Consistency Fields

Yeji Song[1] Chaerin Kong[1] Seoyoung Lee[1] Nojun Kwak*[1] Joonseok Lee*[1,2]
*Corresponding authors [1]Seoul National University [2]Google Research

BMVC 2022

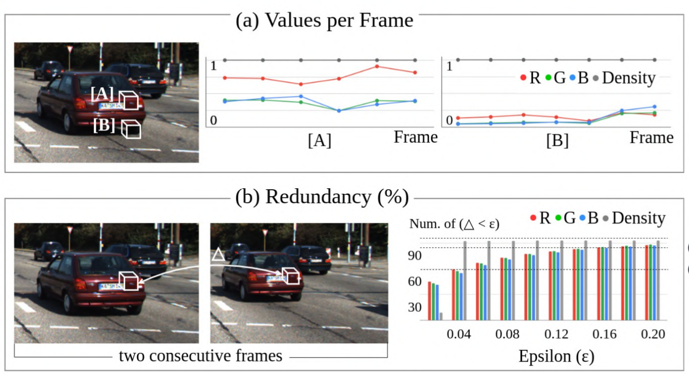
Contributions

NSG Ours (only 15% queries)

- We propose a novel method that **reuses features** which share similar characteristics across image frames.
- We analyze **entangled representations of NSG** that impose potential restrictions in its applications and hinder the feature-reusing framework.
- We empirically demonstrate that our model greatly improves efficiency, **saving up to 85% forward computational cost** with no distinguishable compromise in image quality.

Motivation

Most objects in a video do not significantly change in adjacent frames, then *reuse* redundant features!

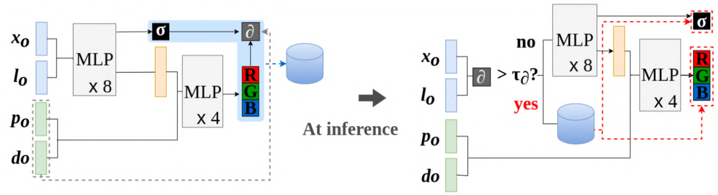


70% bins are this close 90% bins are this close

ε = 0 ❶ ε = 0.04, PSNR= 35.22 ❷ ε = 0.18, PSNR= 24.74

- Most of bins share highly similar values across frames.
- Actively leveraging this redundancy is not only beneficial but in fact necessary to guide our model to learn dynamic scenes more efficiently.
- we propose a **feature-reusing framework** that stores redundant features across frames during training, then at inference reuses them instead of going through a full forward pass.

First Try: Naive Reusing



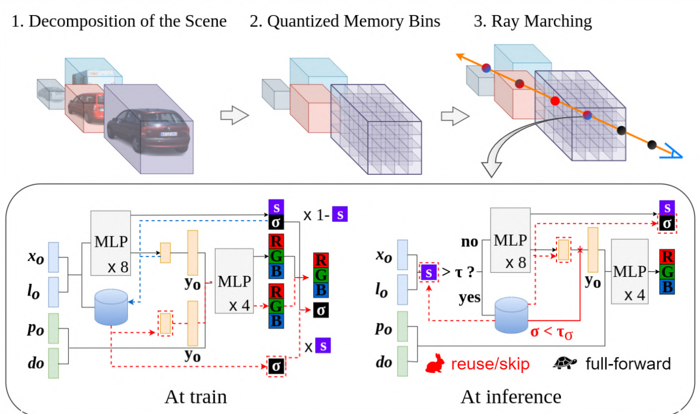
- A bin with a small gradient $\partial = \|\partial\omega/\partial \boldsymbol{p}_o\|_2^2 + \|\partial\omega/\partial \boldsymbol{d}_o\|_2^2$ indicates that values in that bin are most likely not to change across frames.
- For queries in such bins, we directly retrieve the stored values at training.
- However, reusing RGB color values based on gradient norms **generates abnormalities**, since RGB values change abruptly once in a while due to external factors *i.e.*, shadow from nearby environment, global illumination

(a) Reproduction (70% queries) (b) Reproduction of Background (70% queries)

Consistency-Field-based NSG

- We need to consider *consistency*, the characteristics that strongly show **canonical properties of the object** across frames.

CF-NSG: model that learns and reuses *consistent* features.



- *consistency score* **s** estimates how consistent query is across frames.
- *canonical feature* **y** with a high **s** are stored in the memory bin.

Training Objective

regularization term to penalize low reuse

$$\mathcal{L} = \sum_{\boldsymbol{r}\in\mathcal{R}} \left[\|\hat{C}(\boldsymbol{r}) - C(\boldsymbol{r})\|_2^2 + \|\hat{C}_{\text{mixed}}(\boldsymbol{r}) - C(\boldsymbol{r})\|_2^2 \right] + \sum_{\boldsymbol{x},\boldsymbol{x}_o\in\mathcal{X}} \left[\frac{1}{\|s_{bg}(\boldsymbol{x})\|^2} + \frac{1}{\|s_c(\boldsymbol{x}_o)\|^2} \right] + \frac{1}{v}\|\boldsymbol{l}_o\|_2^2,$$

$$\left(\hat{C}_{\text{mixed}} = \textstyle\sum_{i=0}^{N-1} \omega_{\text{mixed},i} \quad \text{where} \quad \omega_{\text{mixed}} = s\cdot\omega_{\text{reuse}} + (1-s)\cdot\omega_{\text{full}}, \quad \omega\in\{(r,g,b,\sigma)\} \right)$$

- If the model reuses improper features largely affected by transient factors, this loss would increase. Therefore, **s leans to predict the proper reusability** and **y with a high s are to be object-intrinsic** across frames.
- CF-NSG **does not disregard transient factors** since second MLP receives transient inputs along with y that well-represents object-intrinsic properties and can learn transient properties more effectively.

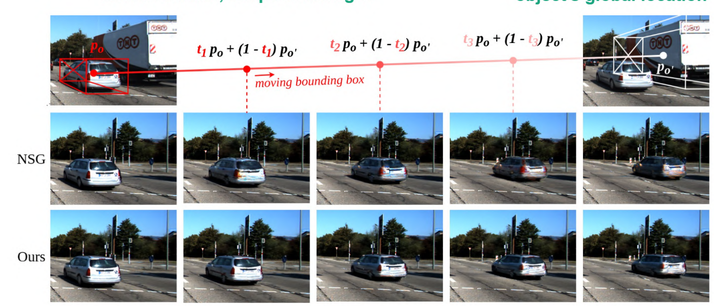
Further Improvements

- Our method enjoys additional gain in efficiency by **skipping a bin with a high consistency score s and a low density** more aggressively.
- Instead of storing y ∈ 256D, we store four factorized 4D vectors and one 256D vector per object, thereby **reducing the memory usage** by 93%.

Experiments and Results

- In CF-NSG, canonical properties do not depend on the external factors.

vehicle's color, shape of tail light **object's global location**



- Comparison results on KITTI

Dataset	Method	#Queries	PSNR(↑)	SSIM [23](↑)	LPIPS [28](↓)	tOF [2]×10^6(↓)	tLP [2]×100(↓)
KITTI	D-NeRF [19]	8.72×	16.33	0.505	0.418	3.823	4.835
	NeRF [12]	7.90×	20.99	0.621	0.446	2.702	3.840
	NSVF [10]	6.38×	22.95	0.706	0.386	2.831	5.071
	NeRF+time [16]	1.64×	24.86	0.653	0.492	2.272	1.563
	NSG [16]	1×	29.54	0.914	0.171	0.619	0.265
	NSG-reduced [16]	0.75×	24.69	0.702	0.452	1.625	1.990
	CF-NSG (ours)	**0.15×**	**28.70**	**0.891**	**0.204**	**0.766**	**0.266**