

Neural Render Proxies for Interactive and Differentiable Lighting

EGSR 2026 - Disney Research

Sergio Sancho, Alexander Rath, Marco Manzi, Pascal Chang,
Amit H. Bermano, Derek Nowrouzezahrai, Markus Gross, Marios Papas

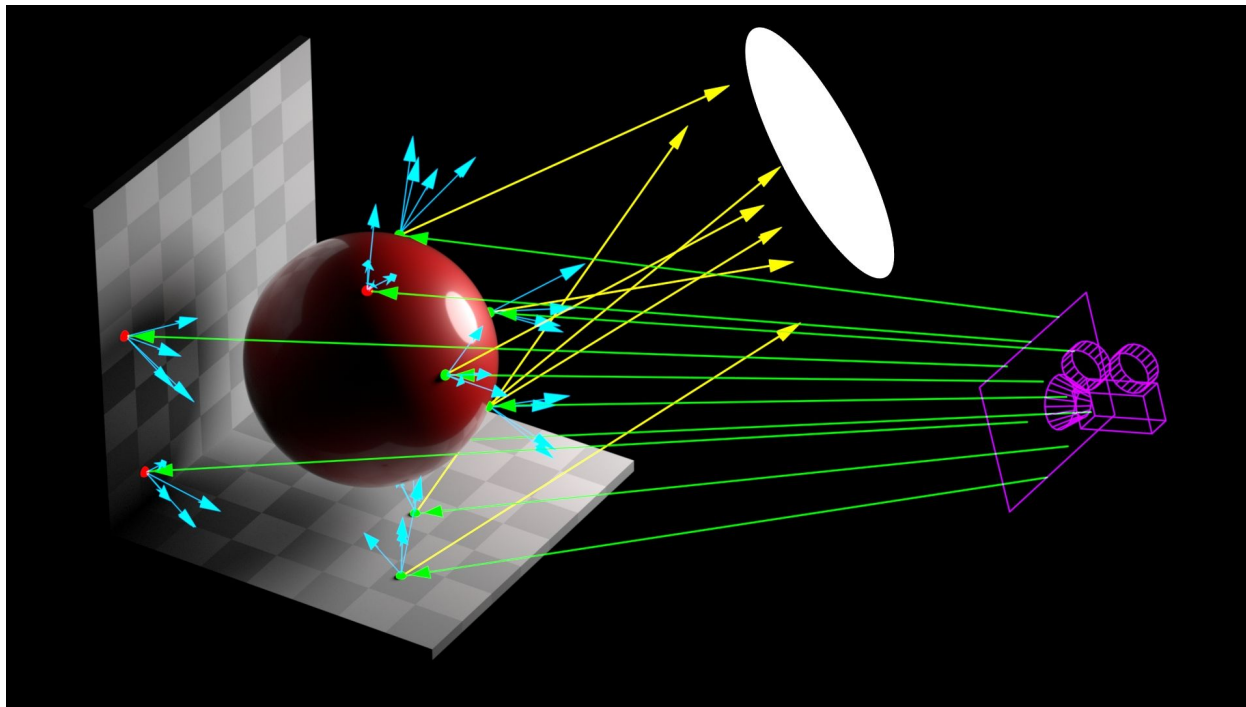
Bastien DOIGNIES - GDL. LIRIS

The Problem: (production) Scene relighting



Time to render a frame: 30m to 1h...

Ray tracing (again...):



The idea:

It is possible to split “path tracing” and “light computations”.

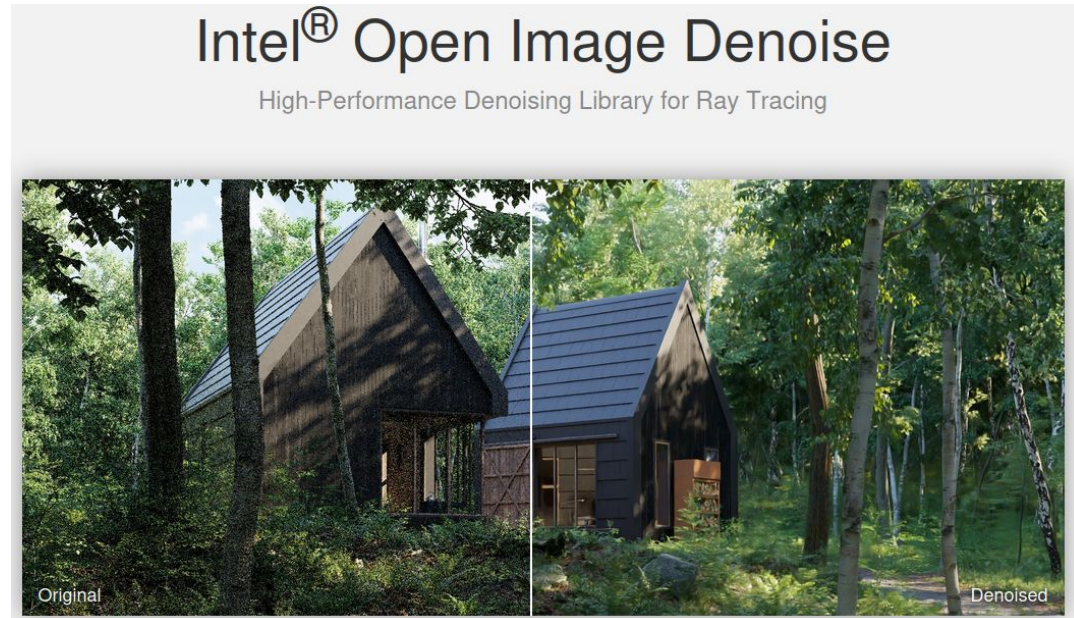
- **First pass:** let the ray bounce around the scene according to the materials (BSDF); and store light throughput weight and hit locations for each ray.
- **Second pass:** replay the paths and compute potential light-source intersections. On hit, use the color attenuation and light throughput weight to propagate back to the pixel.

```
def SAMPLEPATHS(px, scene):  
     $\mathcal{P}_{\text{px}} \leftarrow []$   
    for  $i = 0 \dots N_{\text{spp}}$ :  
         $T_0 \leftarrow \text{RGB}(1, 1, 1)$   
         $x_0, \omega \leftarrow \text{SampleCamera}(\text{pixel})$   
        for  $j = 0 \dots \text{maxDepth}$ :  
             $x_{j+1} \leftarrow \text{Intersect}(\text{scene}, x_j, \omega)$   
             $\omega, f_s, p \leftarrow \text{SampleBsdf}(x_{j+1}, -\omega)$   
             $T_{j+1} \leftarrow T_j \cdot f_s / p$   
             $\mathcal{P}_{\text{px}}.\text{append}([(T_0, x_0), (T_1, x_1), \dots])$   
    return  $\mathcal{P}_{\text{px}}$ 
```

```
def GATHERLIGHT(virtLights,  $\mathcal{P}_{\text{px}}$ ):  
     $I_{\text{px}} \leftarrow \text{RGB}(0, 0, 0)$   
    for  $[(T_0, x_0), (T_1, x_1), \dots] \in \mathcal{P}_{\text{px}}$ :  
        for  $j = 0 \dots \text{maxDepth}$ :  
            for light  $\in \text{virtLights}$ :  
                if hit =  $\text{Intersect}(\text{light}, x_j, x_{j+1})$ :  
                     $I_{\text{px}} \leftarrow I_{\text{px}} + T_j \cdot L_e(\text{light}, \text{hit})$   
    return  $I_{\text{px}}$ 
```

Problems (and their solutions):

- Can not use “light-aware importance sampling” (e.g. NEE, reconnection to light sources).



Problems (and their solutions):

- Storing all paths is costly: “typically weighing dozens of gigabytes per frames”.



Problems (and their solutions):

- Mathematical formulation:

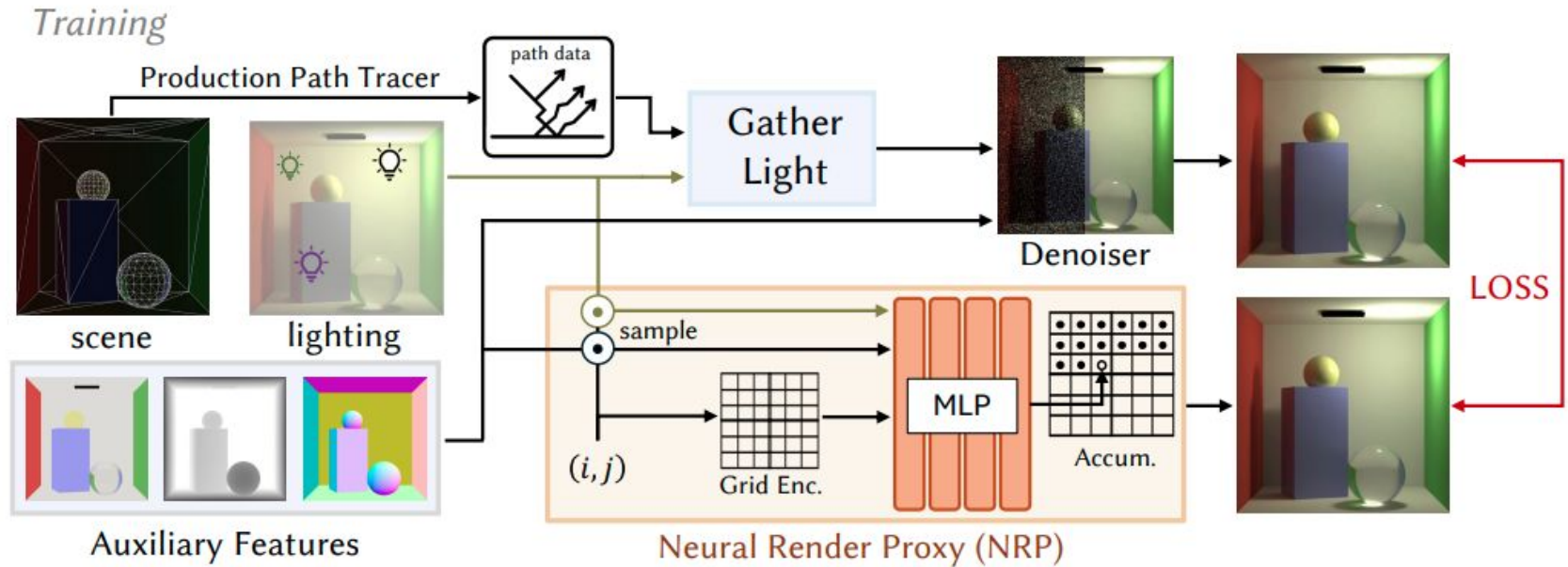
$$\text{GATHERLIGHT}(\mathcal{V}, \mathcal{P}_{\text{px}}) = \sum_{v \in \mathcal{V}} E(v) \cdot \text{GATHER}_{\text{type}(v)}(v, \mathcal{P}_{\text{px}})$$

- Only the Gather part is approximated so that paths do not appear anymore:

$$\mathcal{N}_{\text{type}}(\text{px}, F_{\text{px}}, v) \approx \text{GATHER}_{\text{type}}(v, \mathcal{P}_{\text{px}})$$

Afterward, we can sum the contribution of each light to obtain the final color.

Problems (and their solutions):



Training setup:

- Scene 680x680 512spp.
- MLP “8 layers with 256 neurons”
- 100k it around 1h on an NVIDIA RTX 5090

Reconstruction	32 spp	64 spp	128 spp	256 spp	512 spp	Den.
Time (ms)	1.05	1.86	3.48	6.90	13.89	2.60
GPU Memory (GB)	1.22	2.42	4.82	9.63	19.25	—
Inference	16/4	32/4	64/4	128/8	256/8	512/8
Time (ms)	0.83	0.98	1.20	3.13	9.13	31.21
Model Size (MB)	5.6	5.6	5.7	6.1	7.4	13.3

Results



Inverse rendering



Results



Limitations

- They claim that storing path is prohibitive for animated scenes. But they do not show what the training looks like in this case.
 - Can path be reused?
 - Do we need training for each frame?
- Training is per scene (mlp weights encodes light paths)
- No NEE: When lights are harder to reach at “random” (small, hidden, ...), the method fails.

Limitations



(a) NRP Prediction



(b) Ground-truth Reconstruction

FIN