ARR
Advanced Rendering Research Group

I3D symposium

ACM SIGGRAPH SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES

LSNIF: Locally-Subdivided Neural Intersection Function

2025/05/09

Shin Fujieda

Chih-Chen Kao

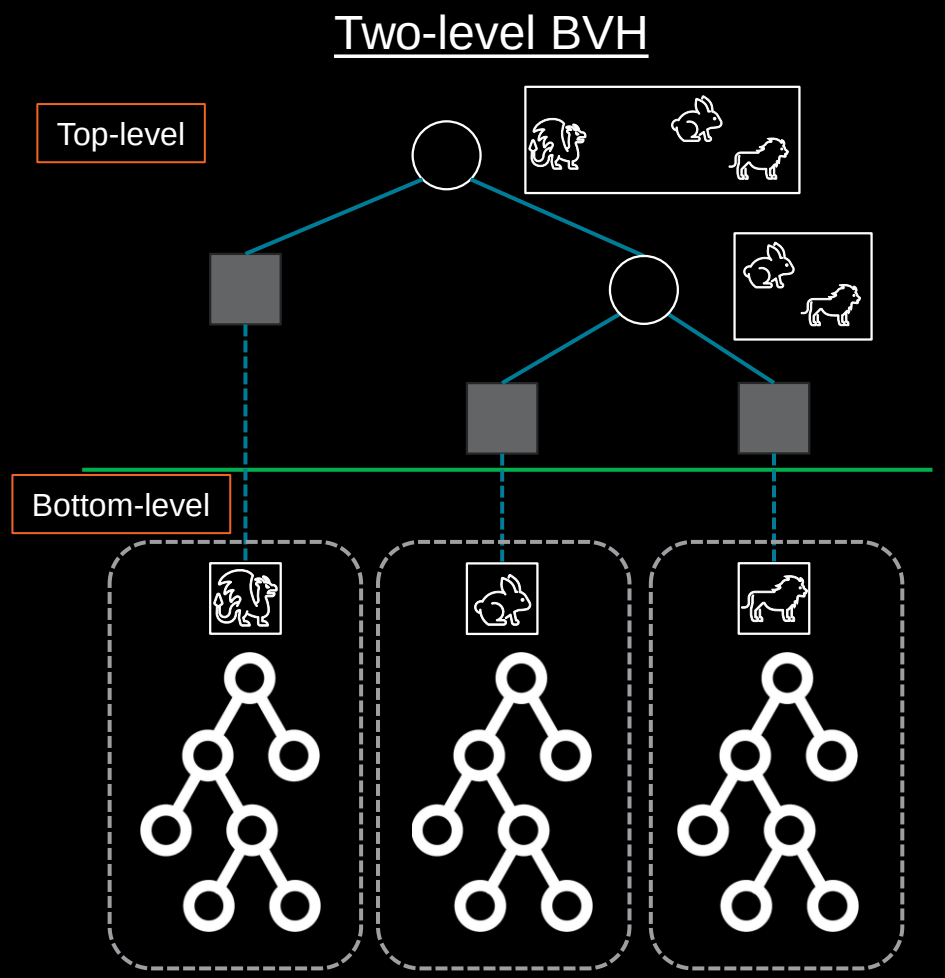
Takahiro Harada

AMD

Introduction

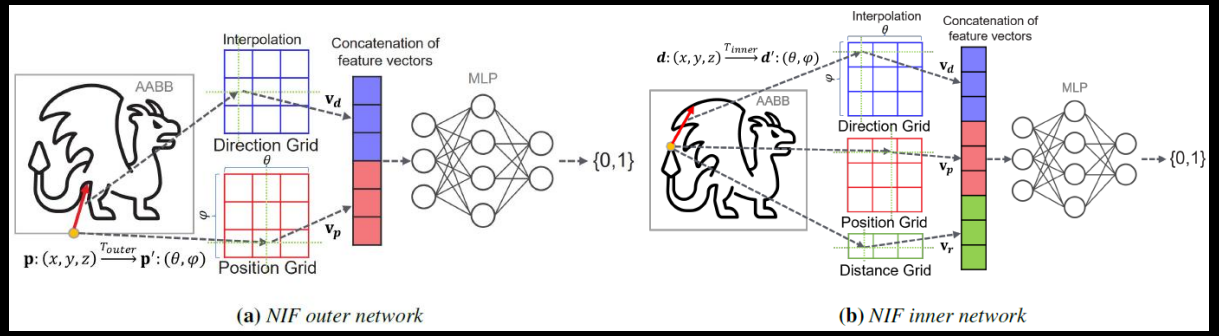
- Conventional Ray Tracing Pipeline - Ray queries
  - Find the ray-geometry intersection
  - Accelerated with a bounding volume hierarchy (BVH)
    - Typically, a two-level BVH

- BVH traversal exhibits irregular execution patterns
  - Branch divergence & unpredictable memory access
  - Hardware cannot fully mitigate high cost of the workload

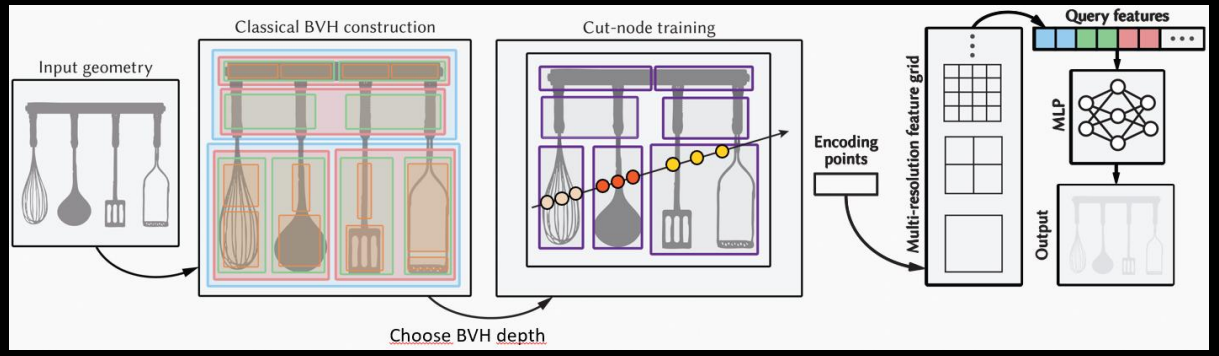


Previous Works with Neural Networks

1. Neural Intersection Function [Fujieda et al. 2023]
   - ✓ Replaces bottom-level BVHs with NNs
   - ✓ Fast and accurate for shadow-ray queries
   - ✗ Needs Online training (viewpoint and lighting)
   - ✗ Primarily limited to shadow-ray usage

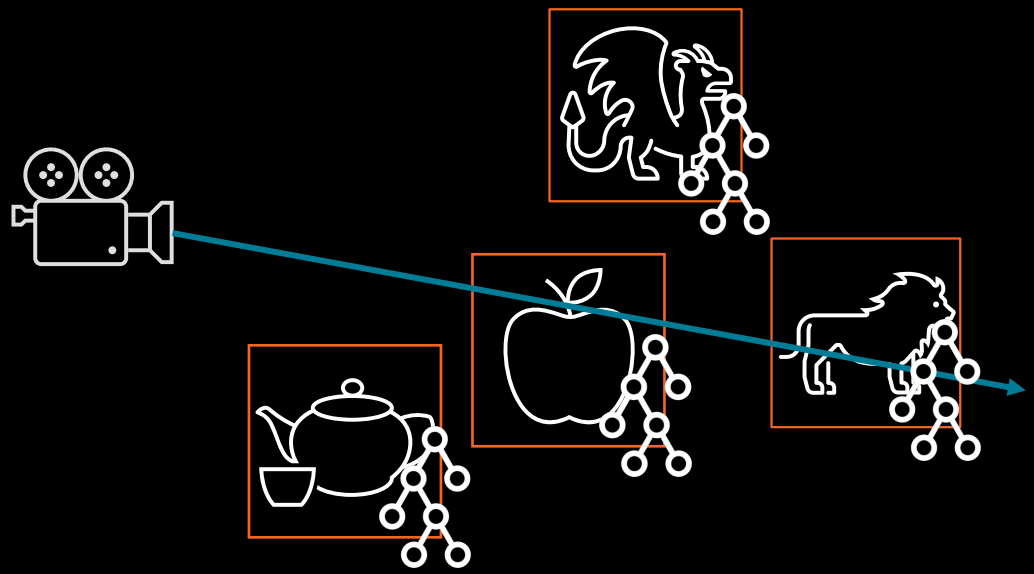


2. N-BVH [Weier et al. 2024]
   - ✓ Offline training
   - ✓ Support any types of ray queries
   - ✗ Overfitted to each scene
   - ✗ No support for dynamic contents

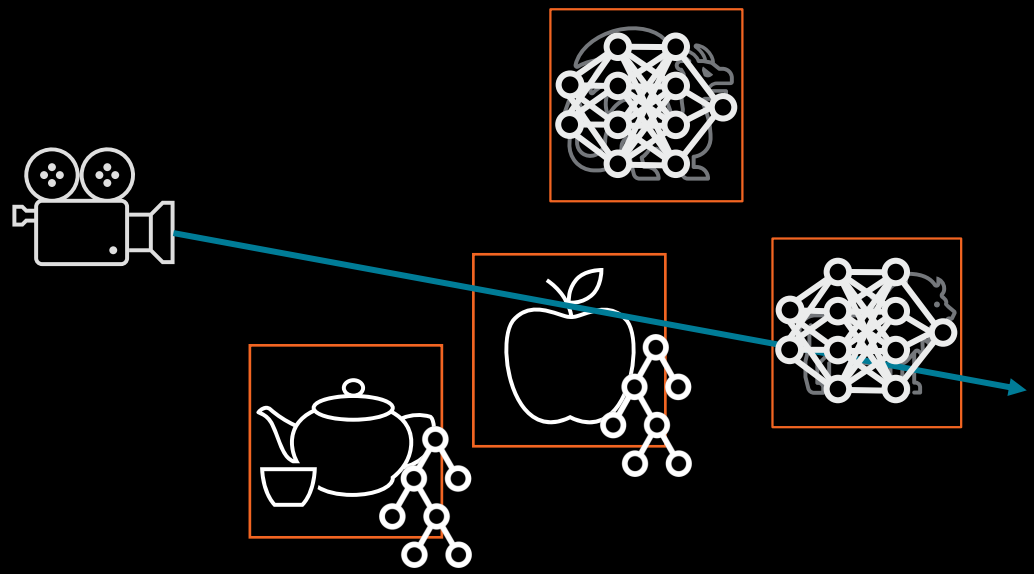


Locally-Subdivided Neural Intersection Function

- Extend NIF to address the limitations of previous works
  - ✓ Offline training for each object, supporting instanced objects
  - ✓ Support scene changes (e.g., camera positions, lighting conditions, rigid transformations)
  - ✓✗ Ray queries for any types of rays other than primary rays



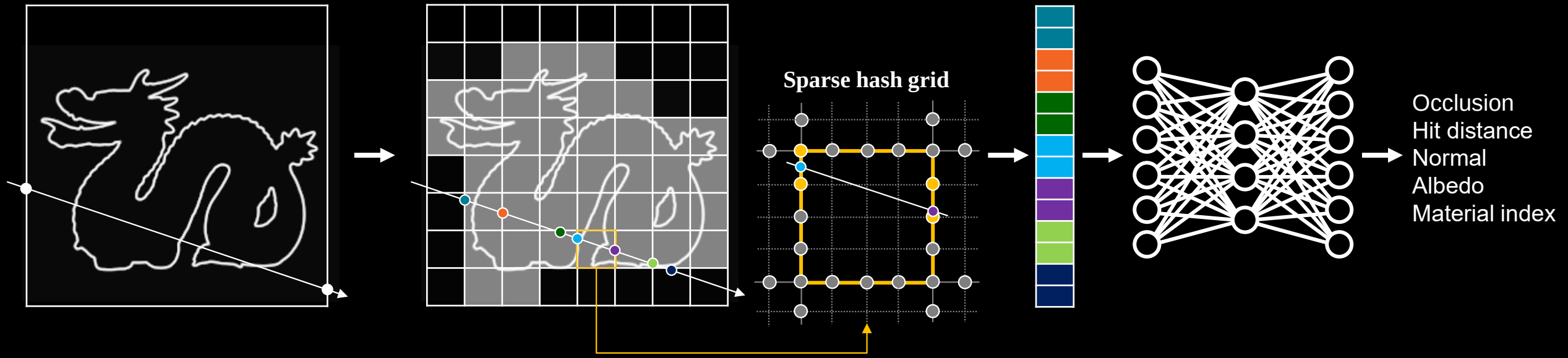
Conventional path tracing
Ray tracing with BVHs



Our hybrid path training
Ray tracing with BVHs and NNs
Complex geometries are replaced with NNs

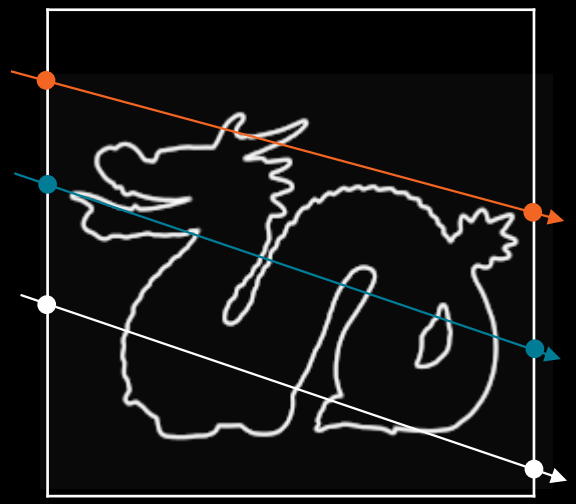
Locally-Subdivided Neural Intersection Function

- Overview of the methodology
- Challenge: each input feature must uniquely represent a specific ray



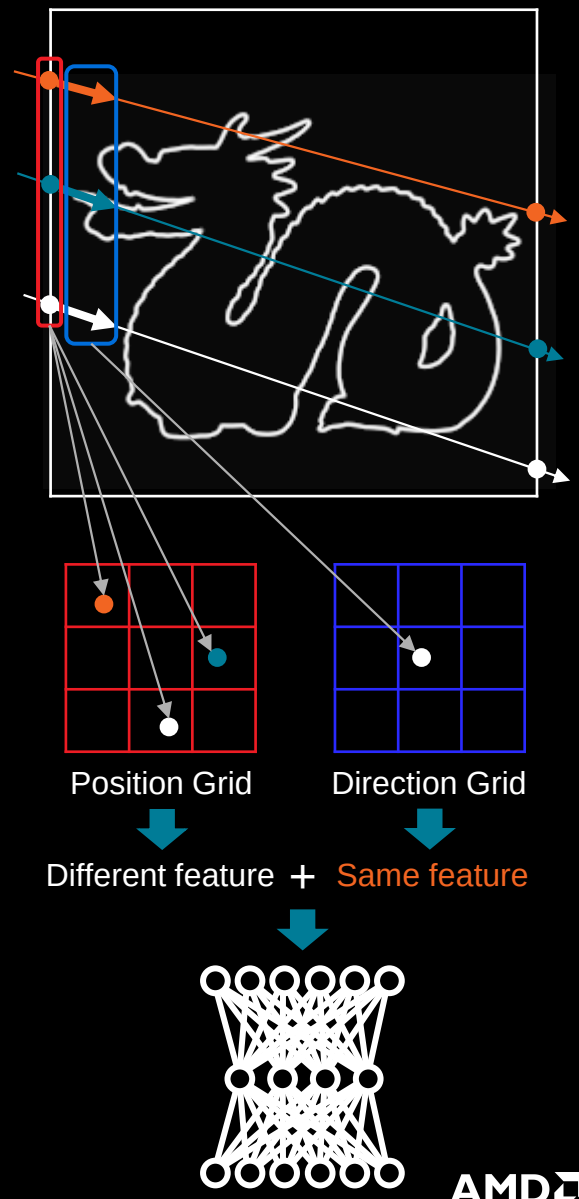
Unique Representation

- Rays with the same direction but originating from different points



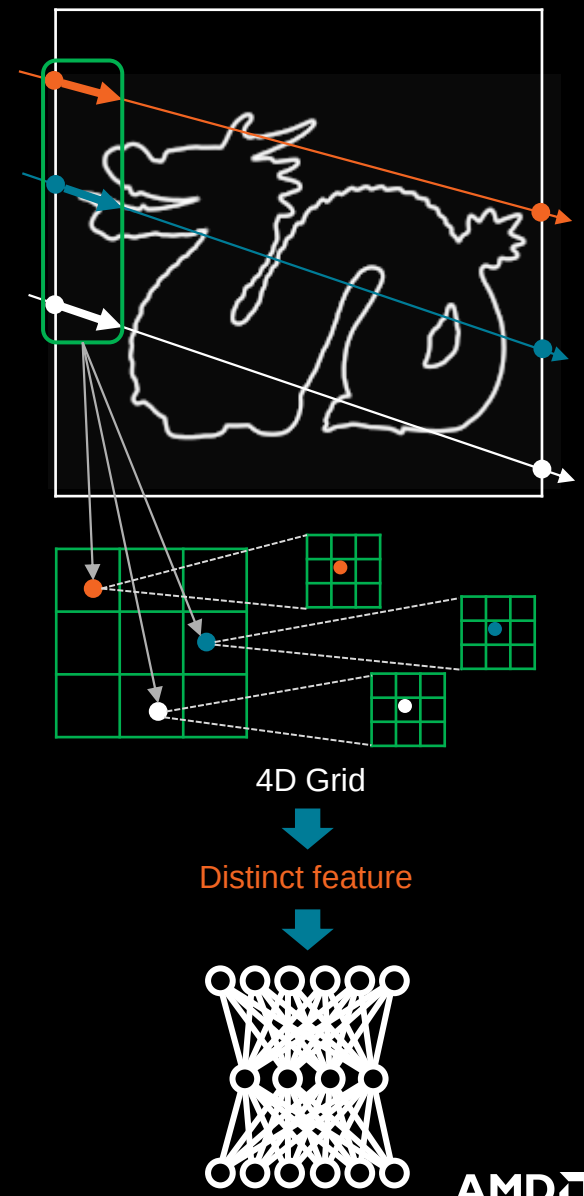
Unique Representation

- Rays with the same direction but originating from different points
- In NIF:
  - NN(Grid2D(Position) + Grid2D(Direction))
  - Effective only when the viewpoint is fixed



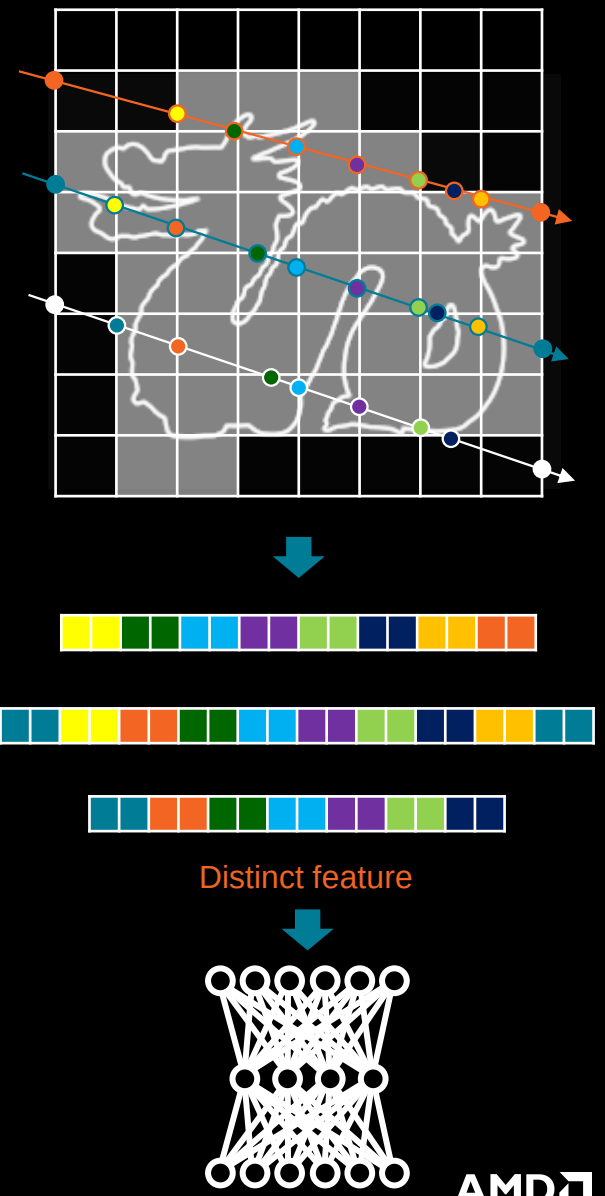
Unique Representation

- Rays with the same direction but originating from different points
- In NIF:
  - NN(Grid2D(Position) + Grid2D(Direction))
  - Effective only when the viewpoint is fixed
- One possible solution:
  - NN(Grid4D(Position, Direction))
  - Computational expensive

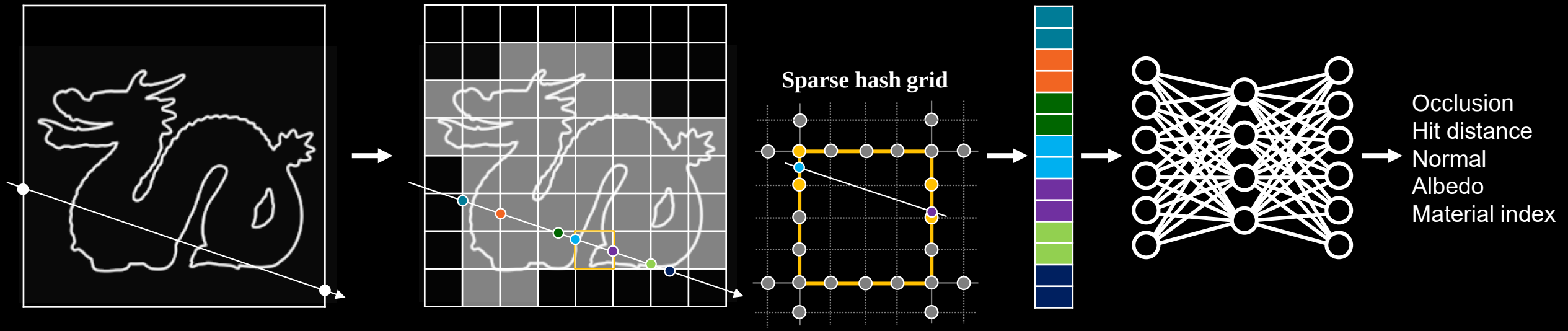


Unique Representation

- Rays with the same direction but originating from different points
- In NIF:
  - NN(Grid2D(Position) + Grid2D(Direction))
  - Effective only when the viewpoint is fixed
- One possible solution:
  - NN(Grid4D(Position, Direction))
  - Computational expensive
- LSNIF solution:
  - Input more data to the MLP
  - Voxelize the AABB to use intersections points as inputs

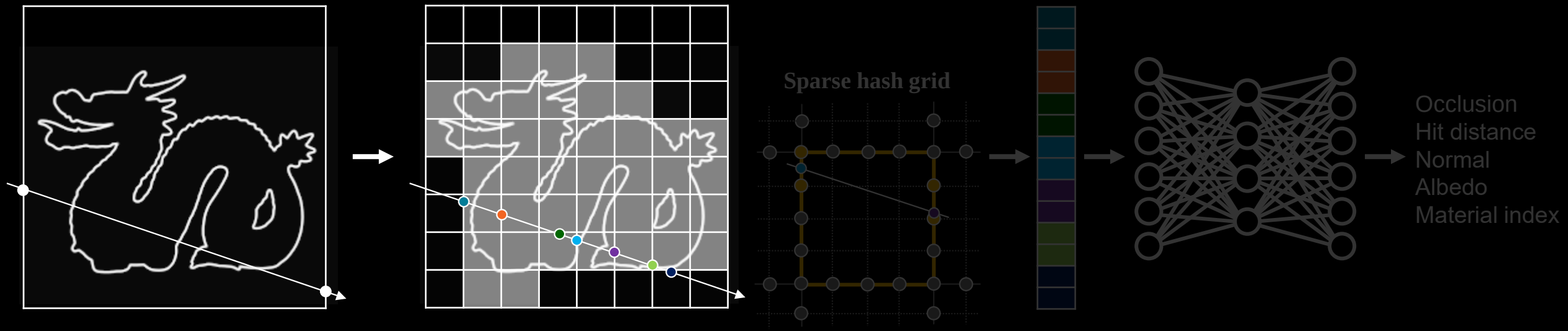


Locally-Subdivided Neural Intersection Function



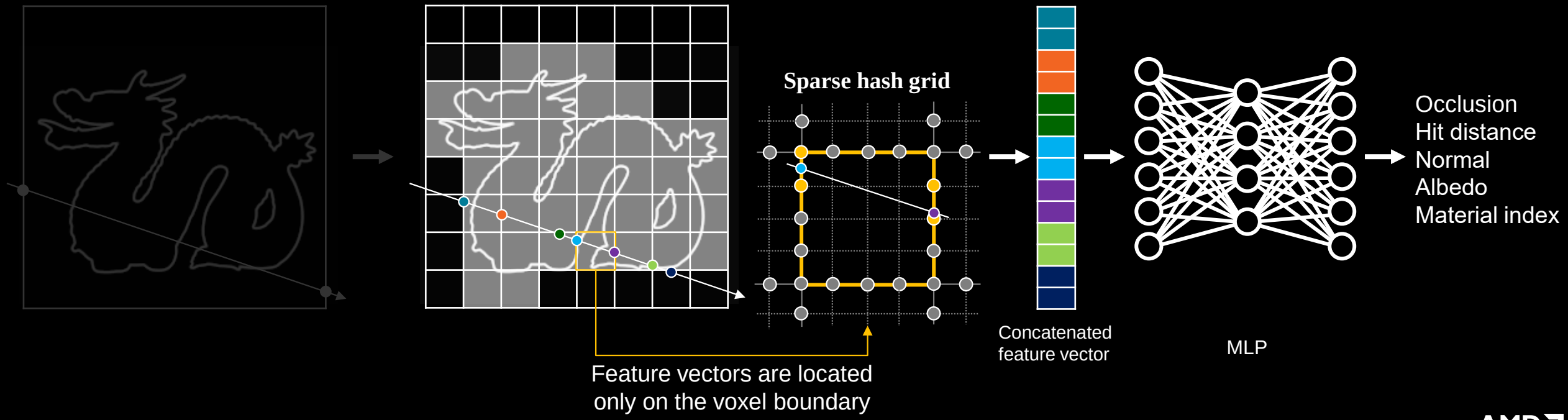
Locally-Subdivided Neural Intersection Function

1. Compute the intersection points of a ray with the AABB
2. Calculate the hit points of the ray on the surface of voxels
   - DDA algorithm based on the intersection points on the AABB



Locally-Subdivided Neural Intersection Function

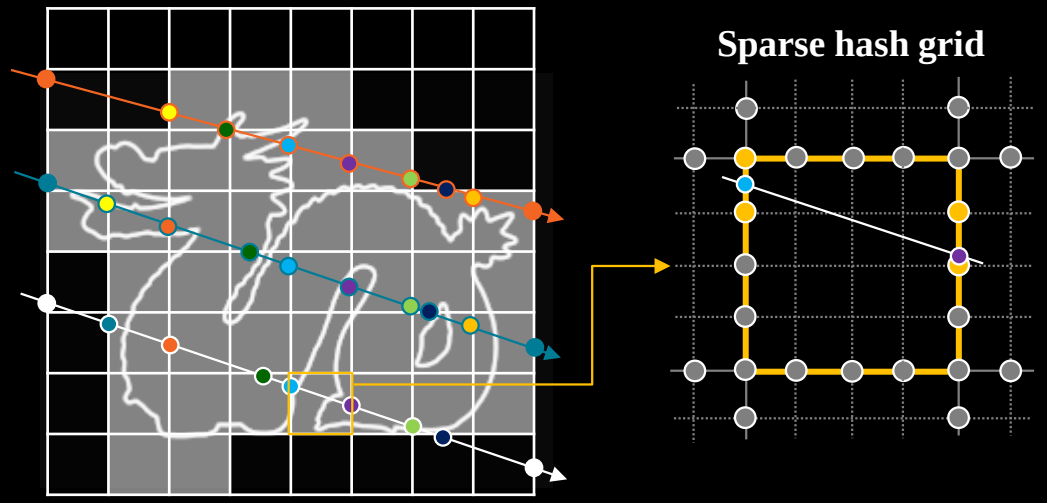
3. Encode the intersections with a multi-resolution sparse hash grid
4. Get the concatenated feature vector
5. Feed the feature vector to an MLP (Multilayer Perceptron) to predict geometric properties



Key Benefits of Voxelization and Encoding in LSNIF

1. Voxelization
   - Distinguish each input that carry different characteristics
   - No explicit voxel occupancy checks compared to sampling within regular intervals

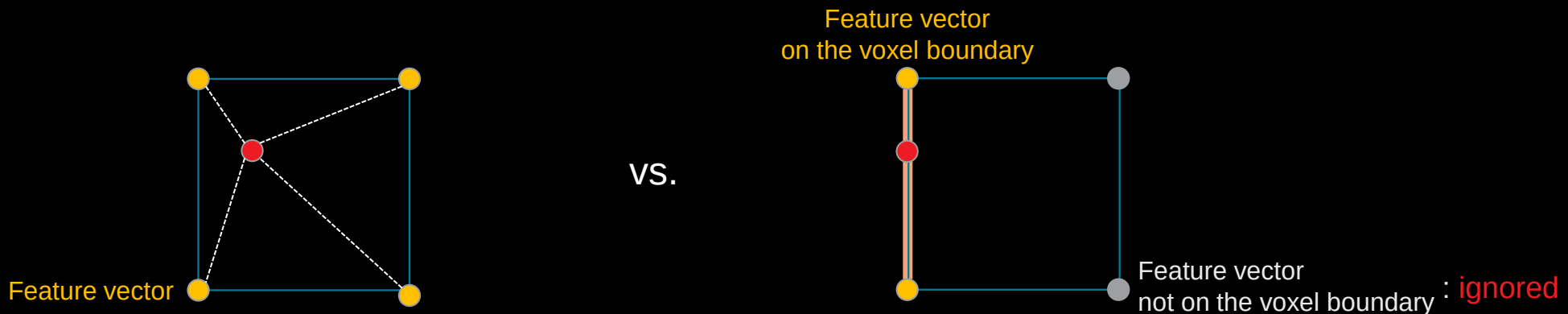
2. Multi-resolution Sparse Hash Grid Encoding



Key Benefits of Voxelization and Encoding in LSNIF

1. Voxelization
   - Distinguish each input that carry different characteristics
   - No explicit voxel occupancy checks compared to sampling within regular intervals

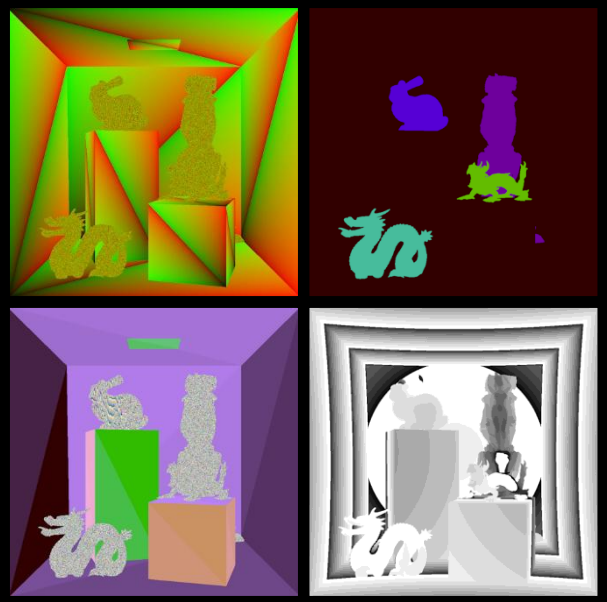
2. Multi-resolution Sparse Hash Grid Encoding
   - Reduce memory footprint: all features are located on the voxel boundaries
   - Reduce number of queries: improve inference performance



2D "Standard" Grid
Query 4 times for bilinear interpolation

2D Sparse Grid
Query 2 times for linear interpolation

The Pipeline of LSNIF

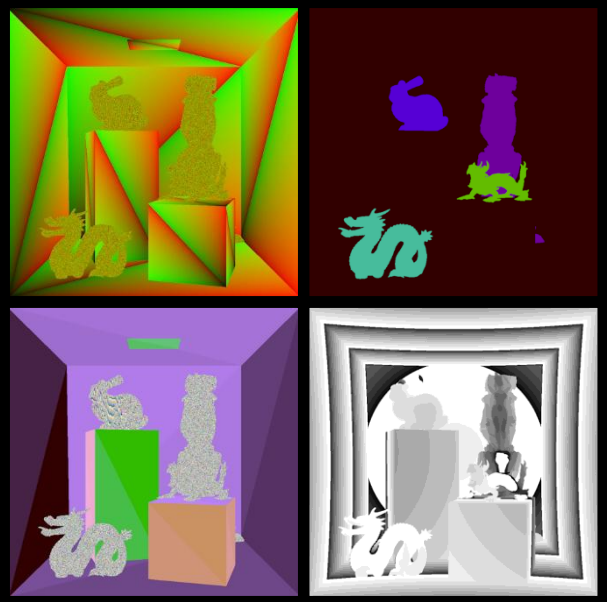


(1) Prepare G-Buffers

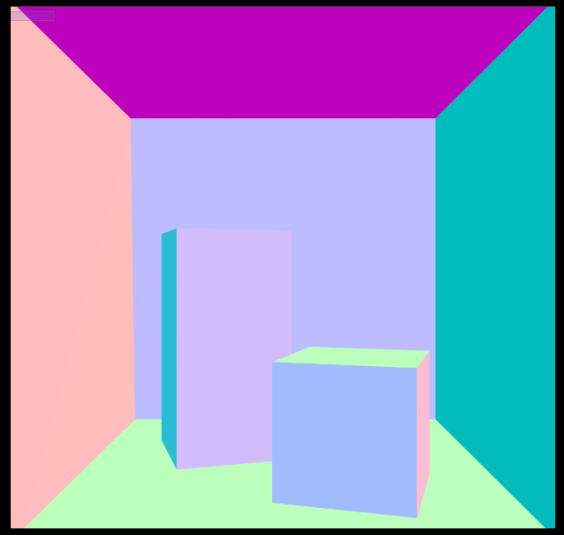
Primary rays

- LSNIF leads to large visual errors
  - Only for non-primary rays
- Use rasterization
  - Store vertex and face information
  - No need for BVH

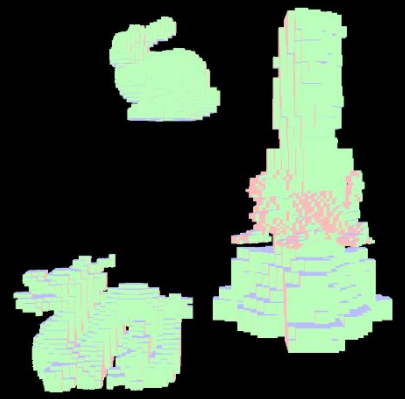
The Pipeline of LSNIF



(1) Prepare G-Buffers



(2) Ray trace using BVH



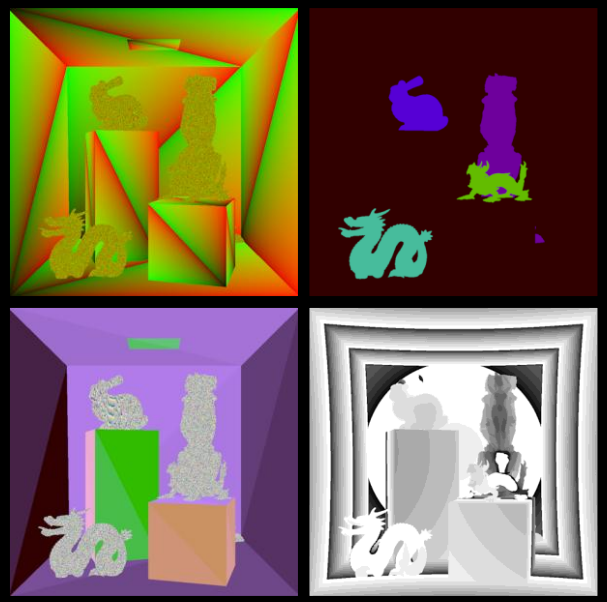
(3) Ray trace using LSNIF

Non-primary rays

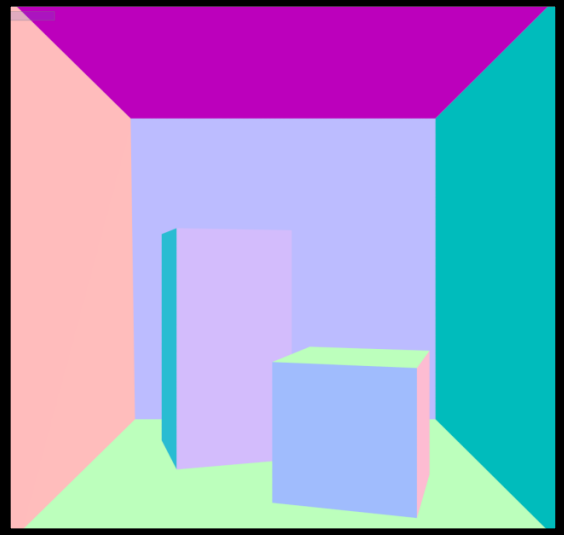
- Simple geometries
  - Two-level BVH
- Shallow and small BVH

- Complex Geometries
  - LSNIF
  - DDA + Sparse grid + Inference
- Dedicated top-level BVH
- Two-phase approach
  1. Collect hits b/w ray and LSNIF objects
  2. Execute LSNIF

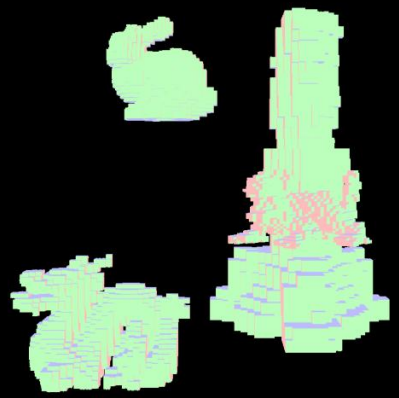
The Pipeline of LSNIF



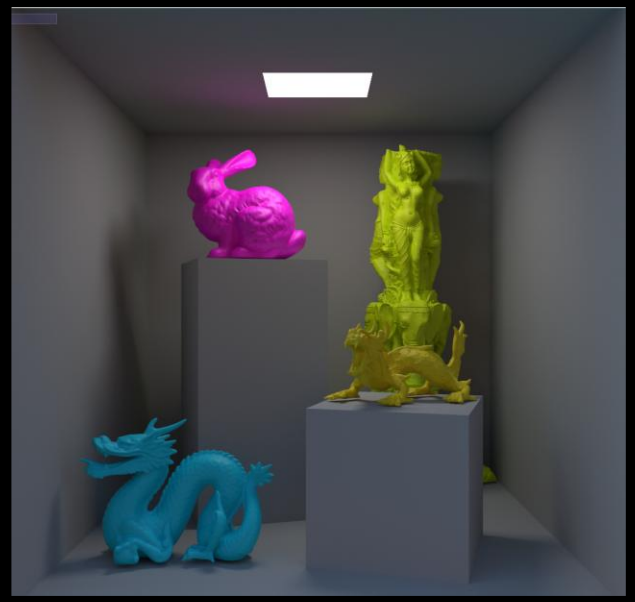
(1) Prepare G-Buffers



(2) Ray trace using BVH

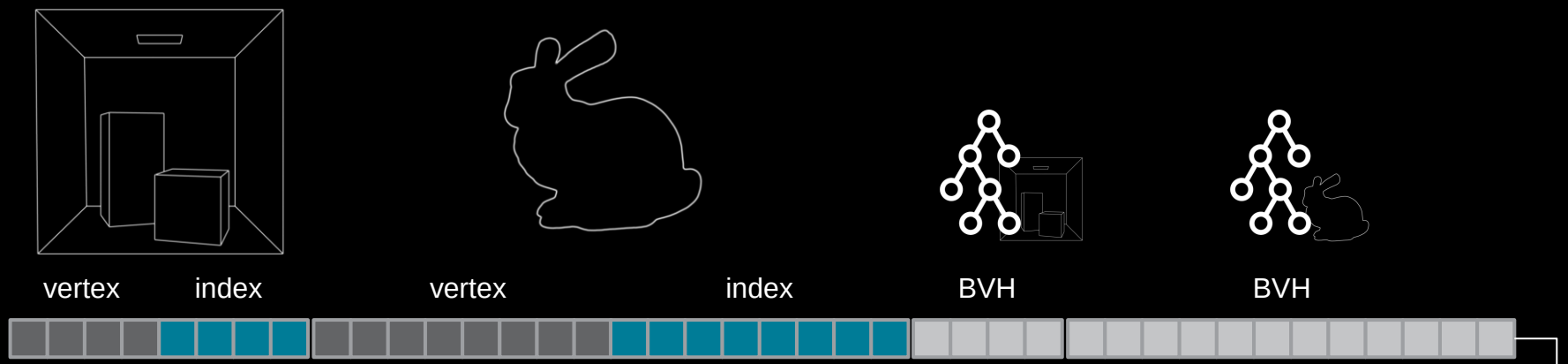


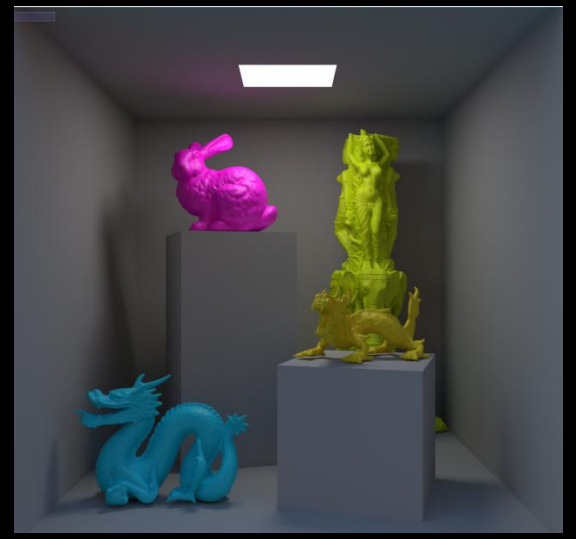
(3) Ray trace using LSNIF

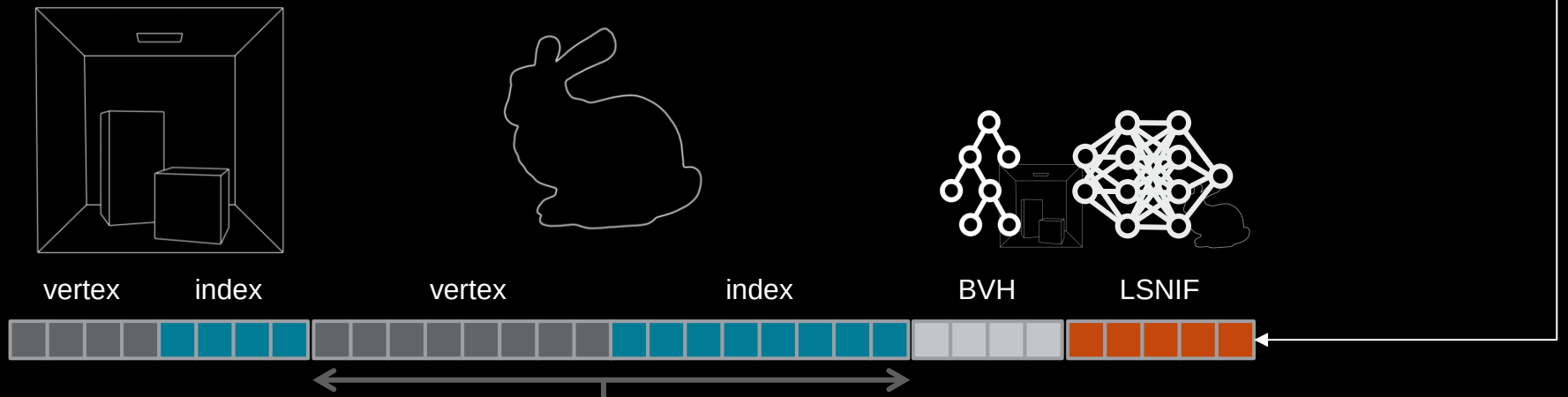


(4) Combine and shade

Comparison between BVH and LSNIF







Up to 106x reduction

This is not needed if we use LSNIF for primary visibility too

Results

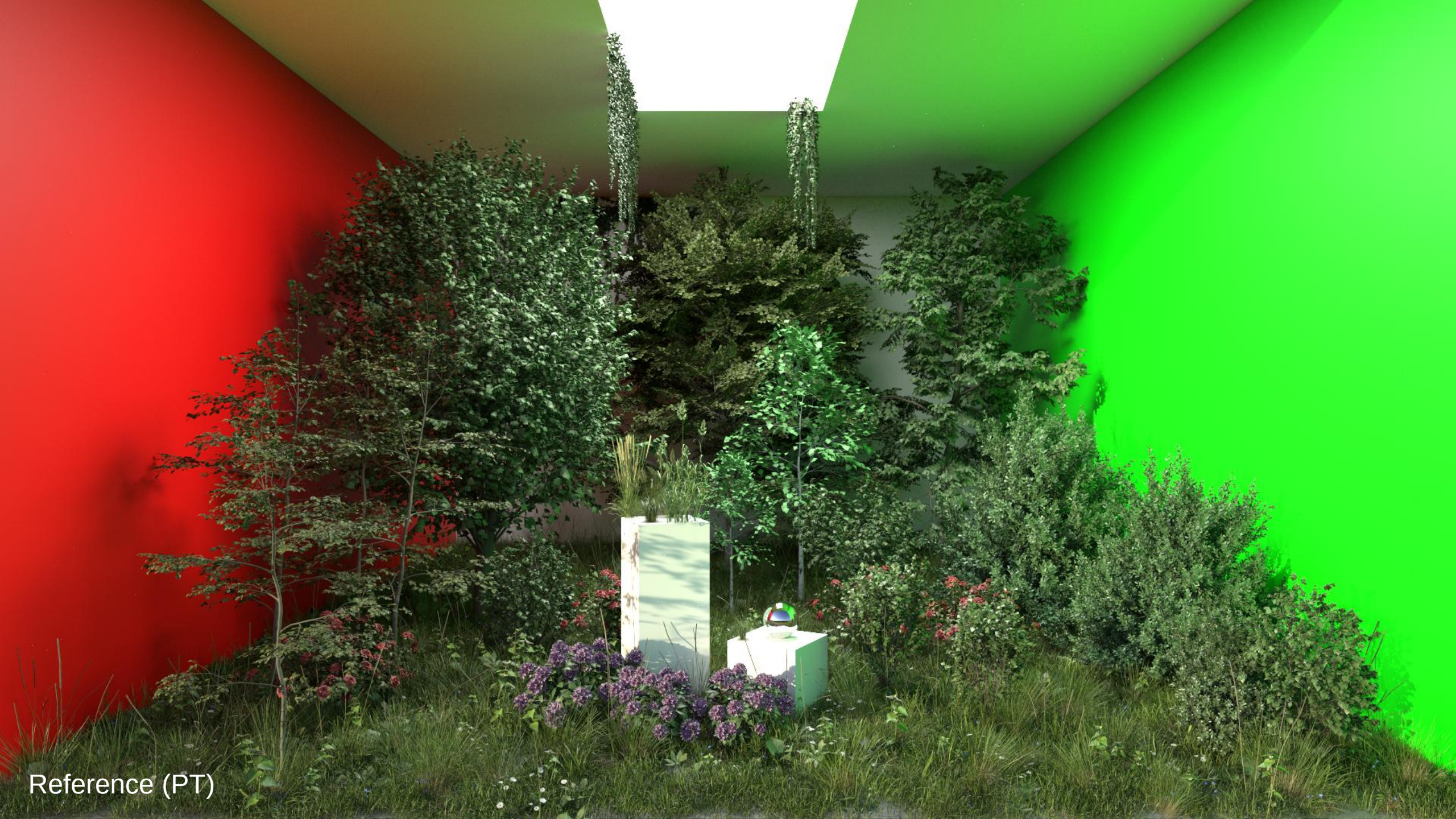
Reference (PT)

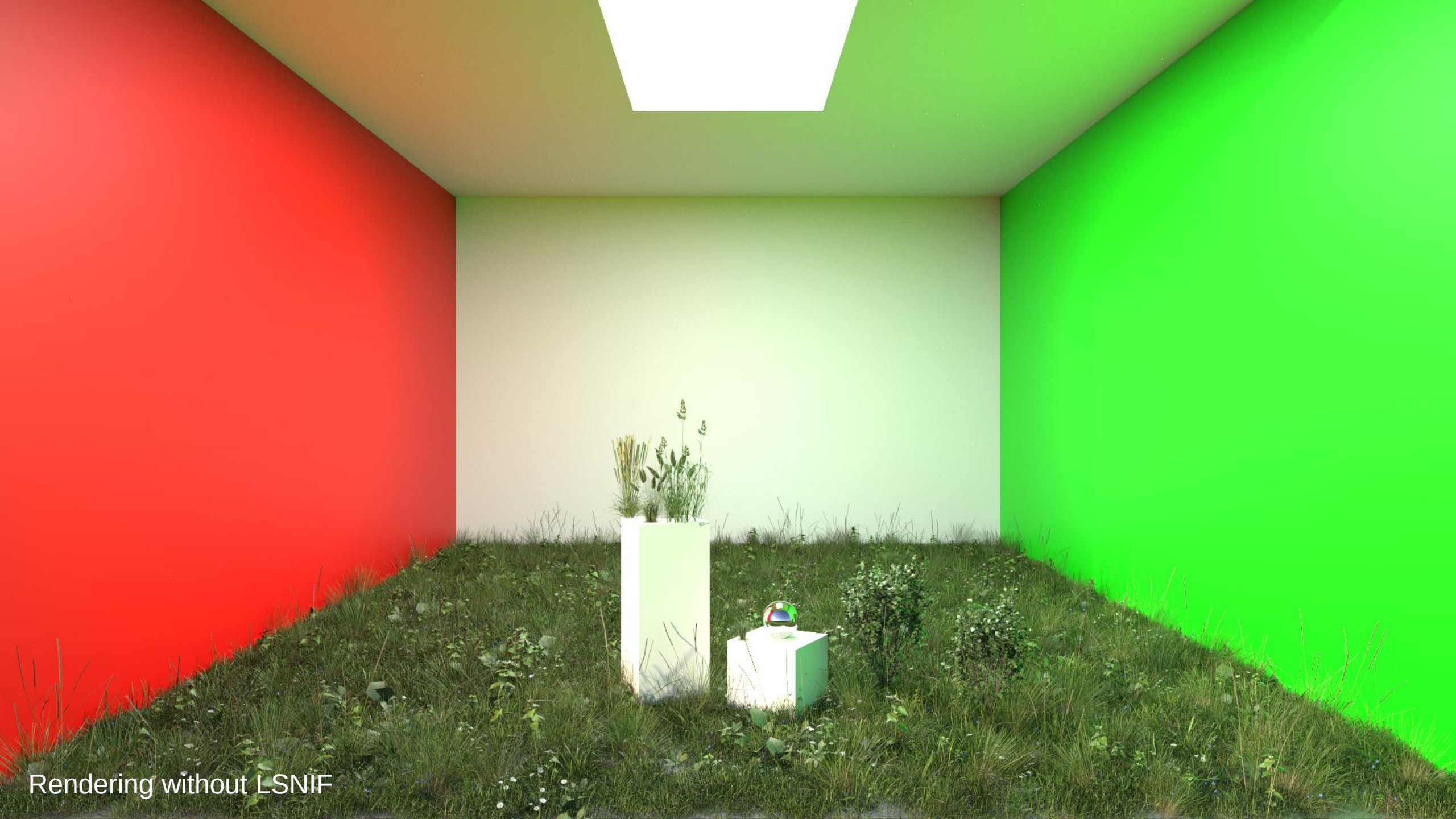
Rendering without LSNIF

Rendering with LSNIF

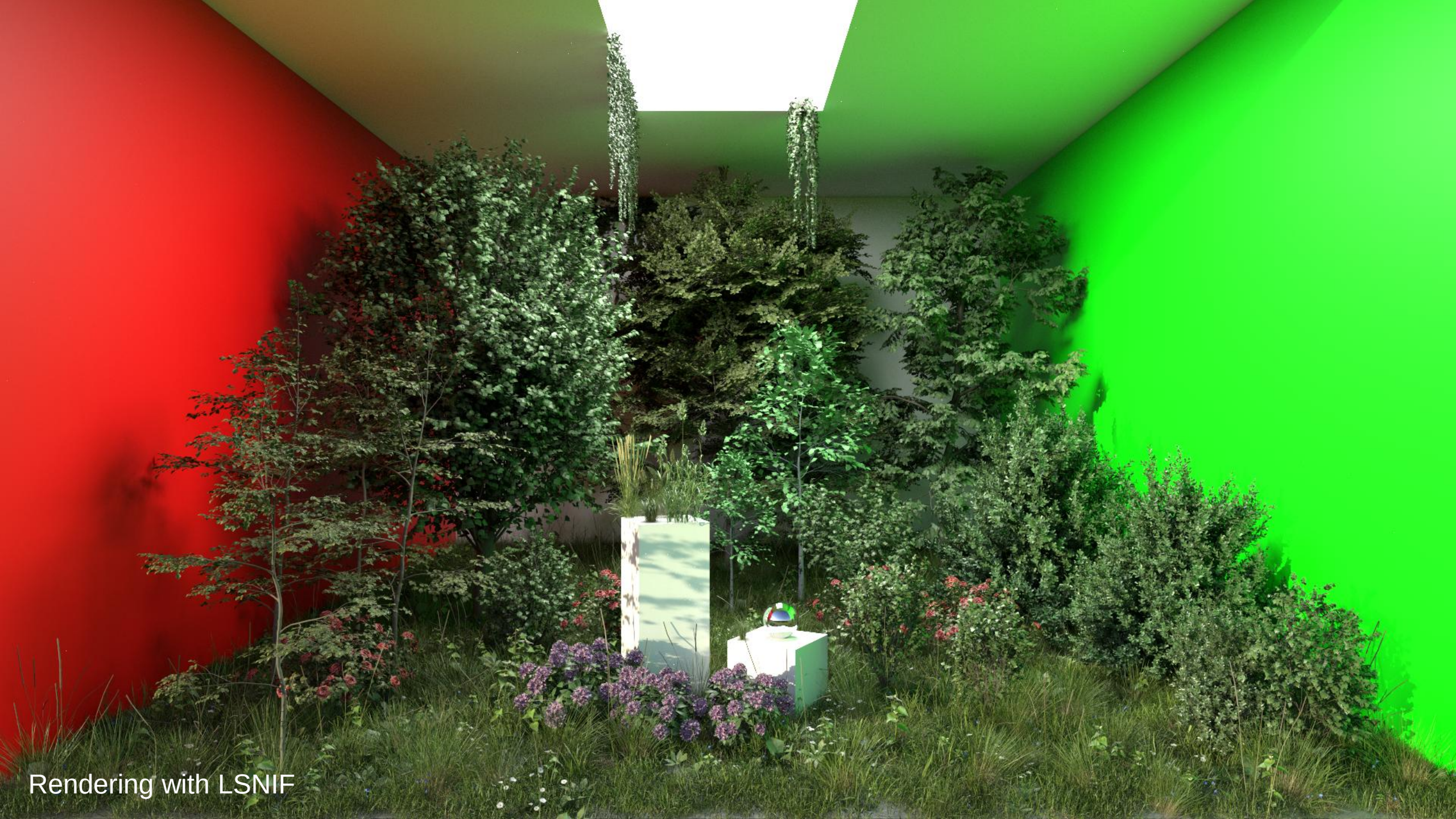
Reference (PT)

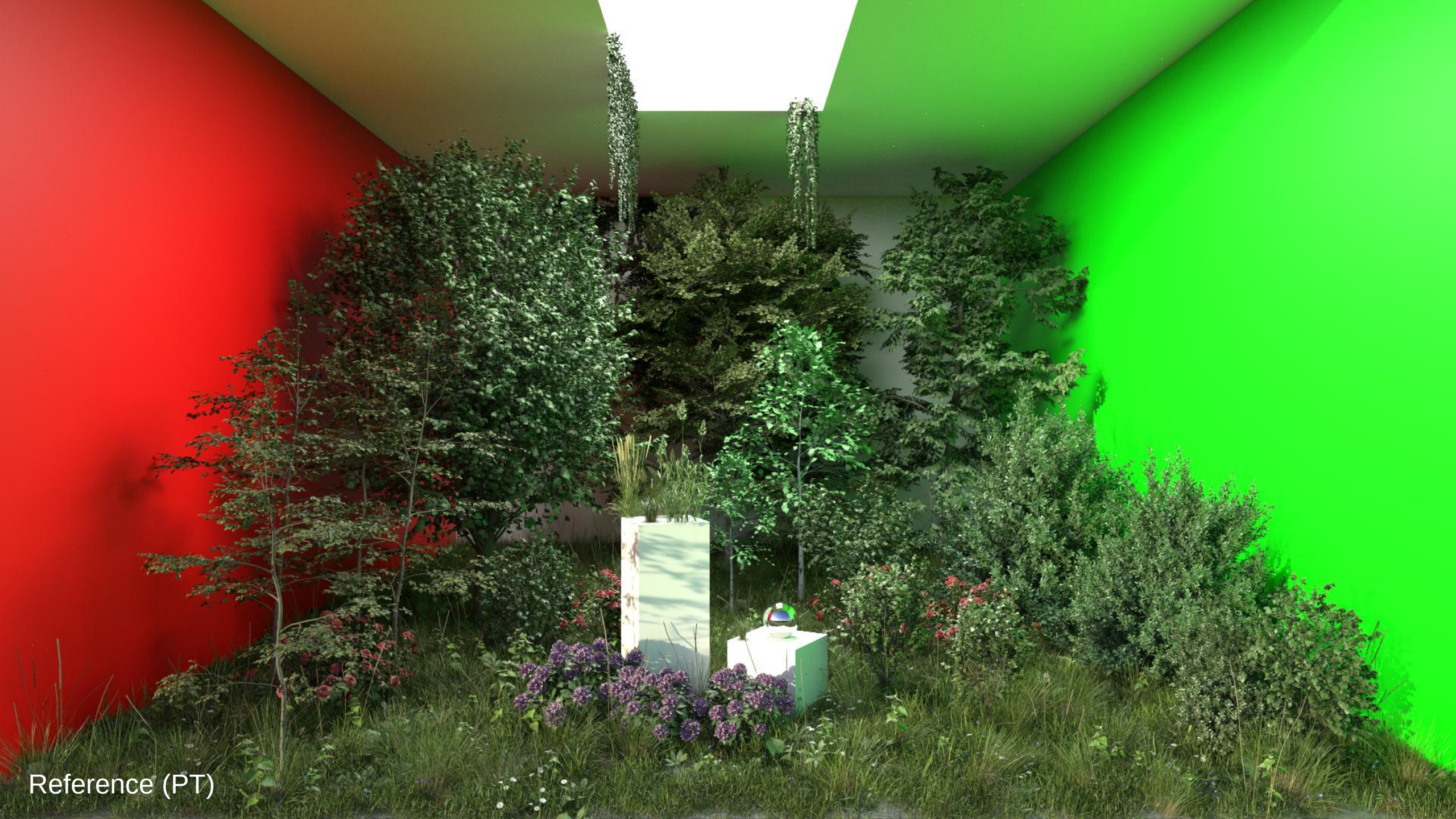

Error


Reference (PT)



Rendering without LSNIF


Rendering with LSNIF


Reference (PT)


Error

Memory Footprint Breakdown

- 1.56 MB / LSNIF
  - 4 KB: 32^3 voxels
  - 62 KB: MLP with two 128-wide hidden layers
  - 1,536 KB: sparse hash grid features with 2 levels of 64^3 and 128^3

- Compressed BVH [Ylitie et al. 2017]
  - 8-wide BVH
  - 80 bytes / node

Compression Ratio & Performance

Scene	LSNIF count	LSNIF	BVH A	BVH B	T^L_{LSNIF}	T^H_{LSNIF}	T_{PT}	$FLIP^L \downarrow$	$FLIP^H \downarrow$
Hairballs	2 (1)	1.56 MB	0.8 GB	0.16 GB	45.56 ms	65.62 ms	84.20 ms	0.115	0.066
5.7M (Tris)		1.00	525.13	106.20	0.54	0.78	1.00		
Dragons	3 (1)	1.56 MB	0.4 GB	0.07 GB	39.12 ms	59.61 ms	36.95 ms	0.049	0.034
2.6M (Tris)		1.00	262.56	48.44	1.06	1.61	1.00		
Statues	4 (4)	6.24 MB	2.5 GB	0.52 GB	37.95 ms	45.15 ms	21.96 ms	0.020	0.021
18.2M (Tris)		1.00	410.26	84.77	1.73	2.06	1.00		
Junkshop	9 (9)	14.04 MB	2.9 GB	0.58 GB	42.27 ms	49.61 ms	39.54 ms	0.035	0.037
20.5M (Tris)		1.00	211.51	42.44	1.07	1.25	1.00		
Botanic Cornell Box	26 (15)	23.40 MB	3.8 GB	0.77 GB	150.31 ms	173.97 ms	162.40 ms	0.142	0.108
27.0M (Tris)		1.00	166.29	33.54	0.93	1.07	1.00		
Botanic Cornell Box	26 (15)	23.40 MB	3.8 GB	0.77 GB	147.35 ms	176.47 ms	158.76 ms	0.198	0.170
Glossy 27.0M (Tris)		1.00	166.29	33.54	0.93	1.11	1.00		
Classroom	50 (15)	23.40 MB	0.4 GB	0.07 GB	53.86 ms	62.01 ms	57.36 ms	0.051	0.036
2.6M (Tris)		1.00	17.50	3.23	0.94	1.08	1.00		

Memory footprint comparison

Rendering time comparison

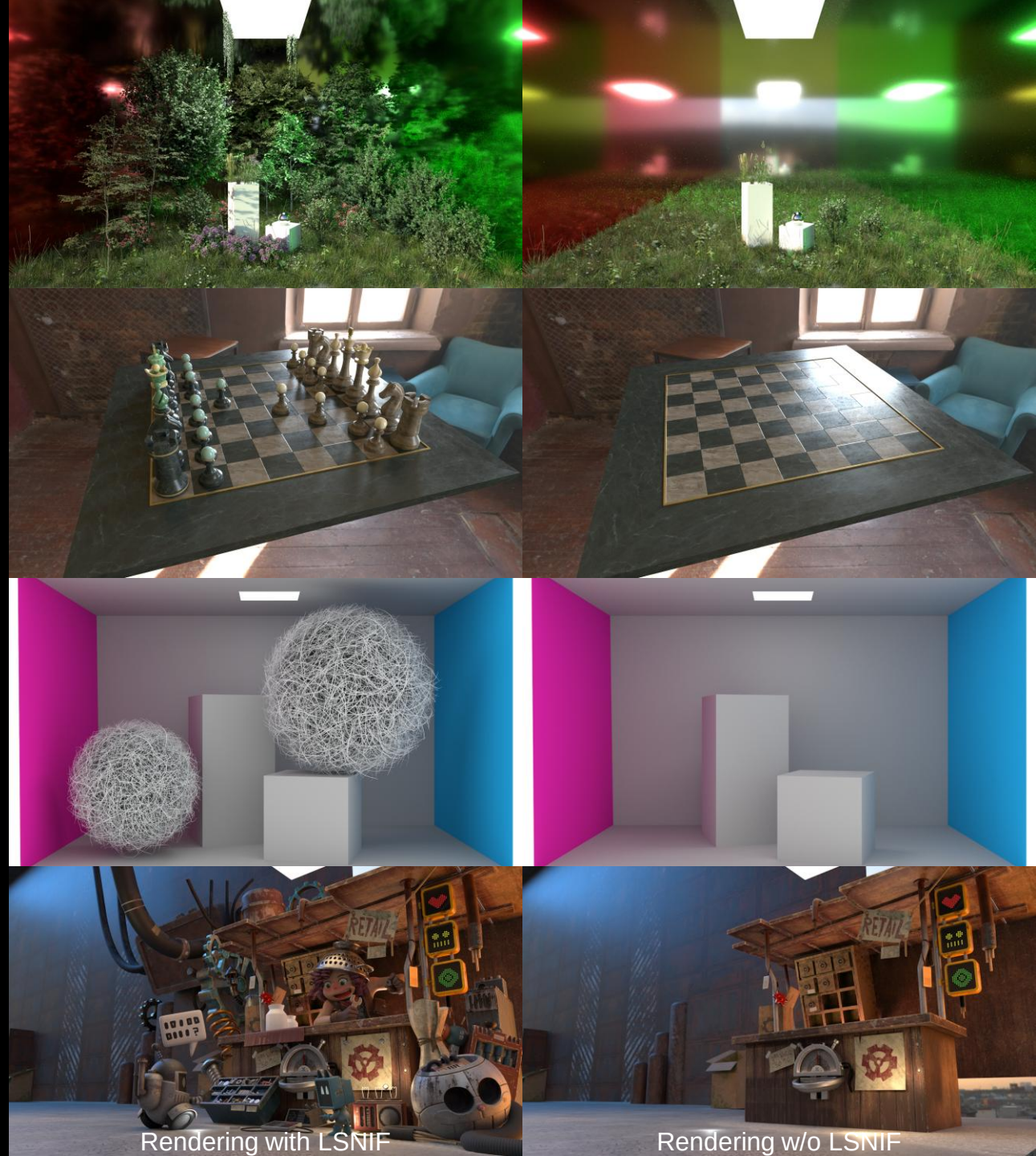
Changing Camera

Limitations and Future Extensions

- Rasterization for primary rays
  - Retain vertex and face information of geometries
  - Higher quality of reconstructed geometries
- Deformation support
  - Require additional training
- Texture support
  - Errors in texture-coordinate prediction directly result in visible artifacts
- LOD support
  - Simple solution is to store multiple LSNIFs for different LODs with different voxel resolutions

LSNIF Summary

- Replace bottom-level BVHs with NNs
- Offline training for each object
- Support limited scene changes
  - Transforms of objects, lights, camera
- Hybrid rendering pipeline uses
  - Rasterization
  - Ray tracing
  - Machine learning
- 106x memory reduction compared to compressed BVH
  - 1.56 MB / LSNIF
- Outperform BVH ray tracing for complex scenes



I3D symposium

AMD

Thank you!

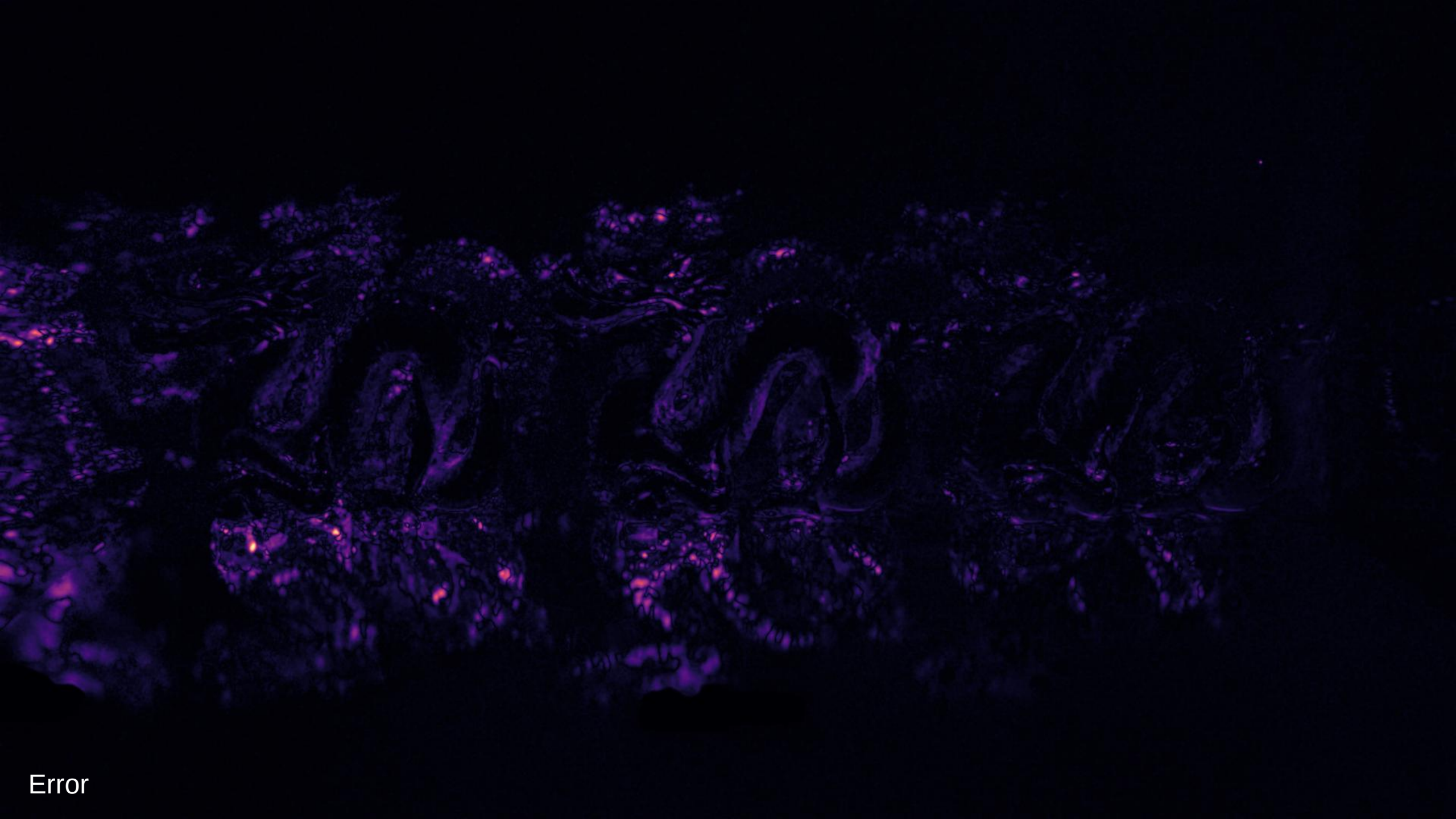
Showing the scene conversion pipeline
Single script execution converts the scene into LSNIF
Junkshop scene conversion finishes within 40s


Reference (PT)

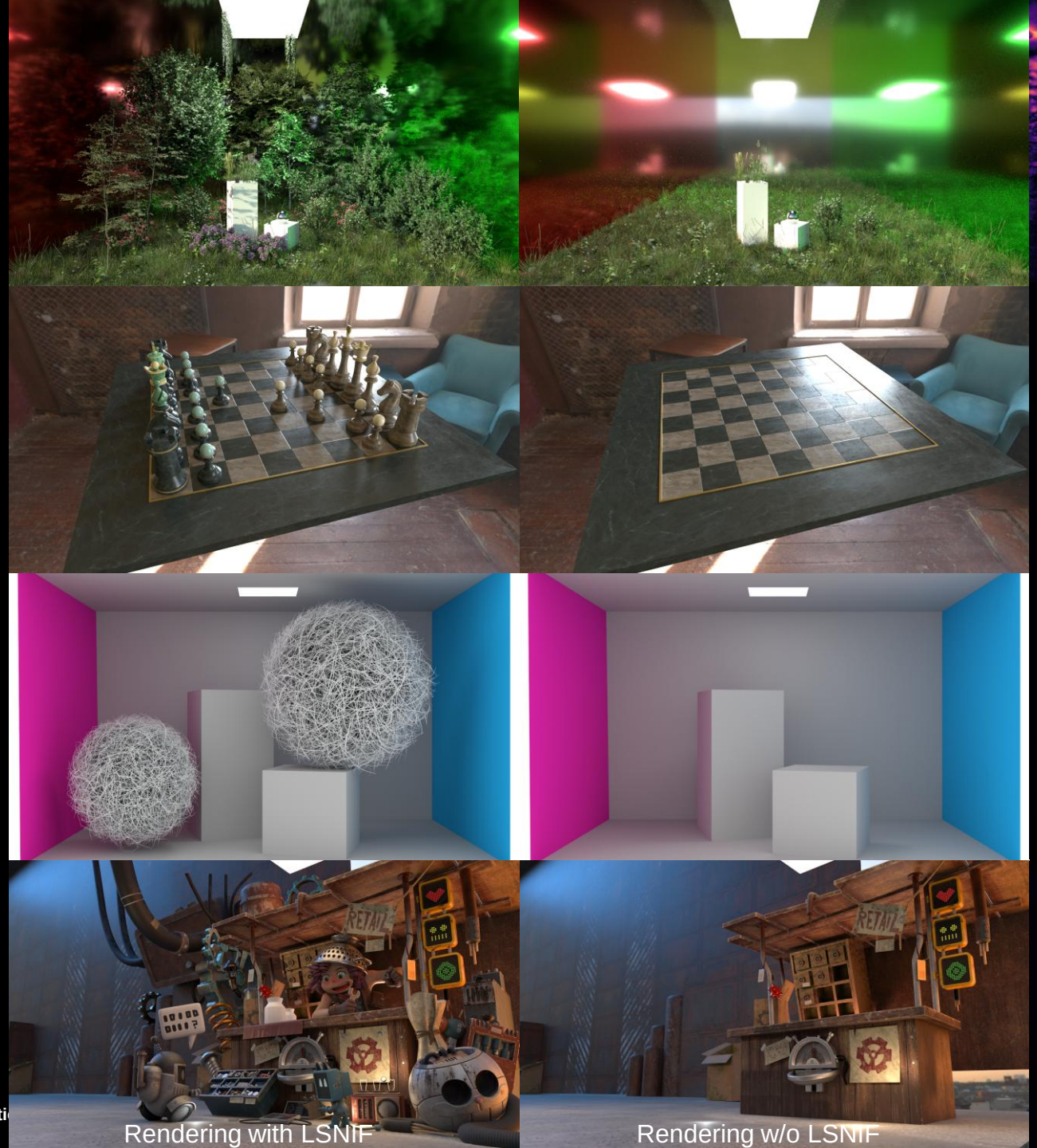
Rendering without LSNIF

Rendering with LSNIF


Reference (PT)

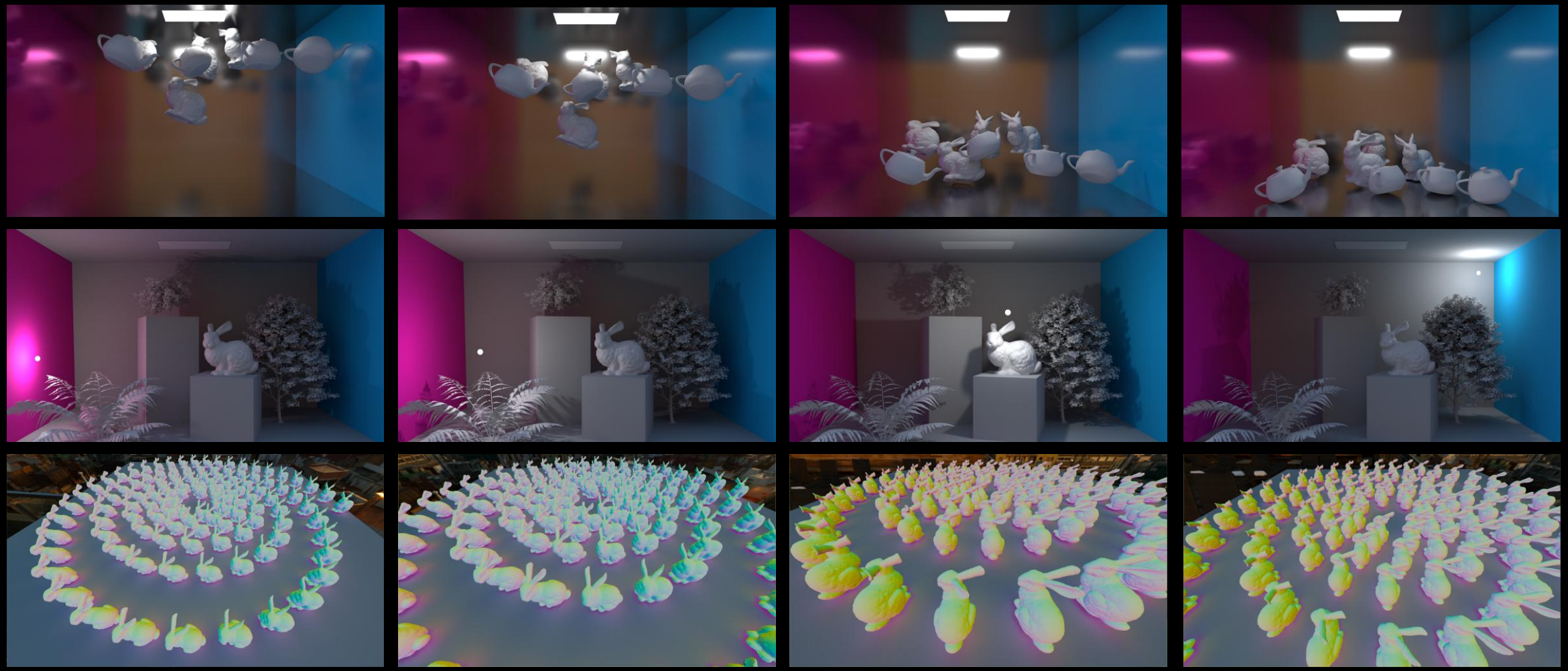

Error

Other Results

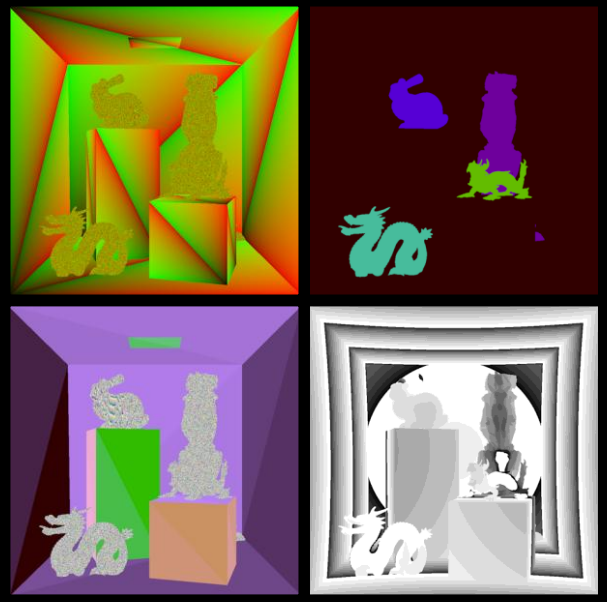


Rendering with LSNIF | Rendering w/o LSNIF | Error

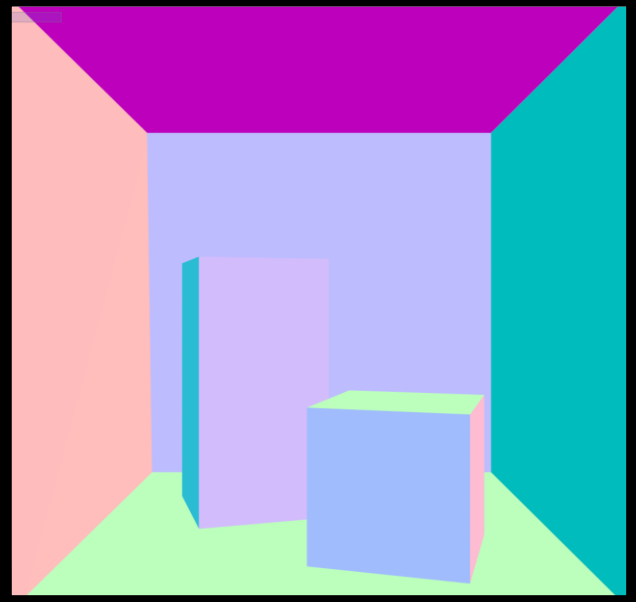
Limited Scene Change Support



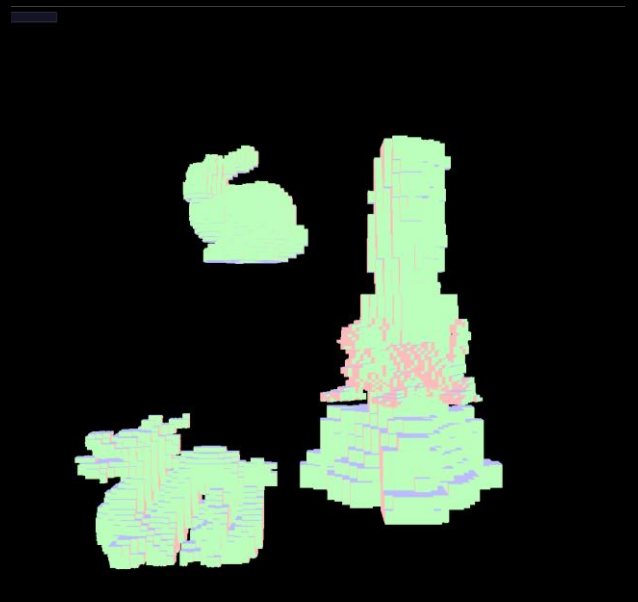
The Pipeline of LSNIF



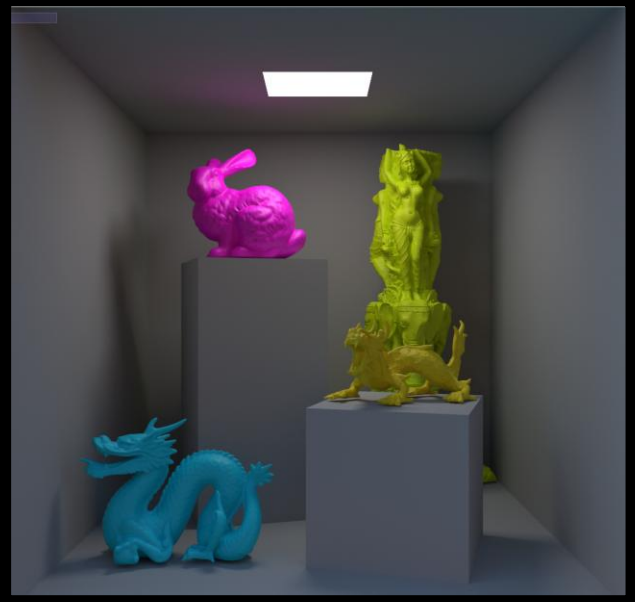
(1) Prepare G-Buffers
(Rasterization)



(2) Ray trace using BVH
only for simple objects



(3) Ray trace using LSNIF
DDA + Sparse Grid Encoding + Inference



(4) Combine and shade

Simple geometry → standard two-level BVH

✚

Complex geometry → dedicated top-level BVH + LSNIF

The Pipeline of LSNIF

Preprocess:

- Complex geometry → LSNIF
- Simple geometry → bottom-level BVH
- Build a top-level BVH on top of those (hybrid BVH)

```
if( node.isLeafNode() )
{
        if( node.leafHasNN() ) // complex geometry
        {
                res = runDDA( theObject ); // new step
                inputVector = applyInputEncoding( res );
                executeNN( inputVector );
        }
        else // simple geometry
        {
                traverse( bottomBVH );
        }
}
```

Runtime:

- Rasterize primary
  - Not store BVH for LSNIF objects
  - Retain face and vertex information
- Trace rays through the hybrid BVH
  - Ray tracing is minimum as it only need to trace against simple geometries (and top-level BVH)
- For NIF objects
  - Run DDA to collect intersecting voxels
  - Input encoding (feature vector look up and concatenation)
  - Execute NN

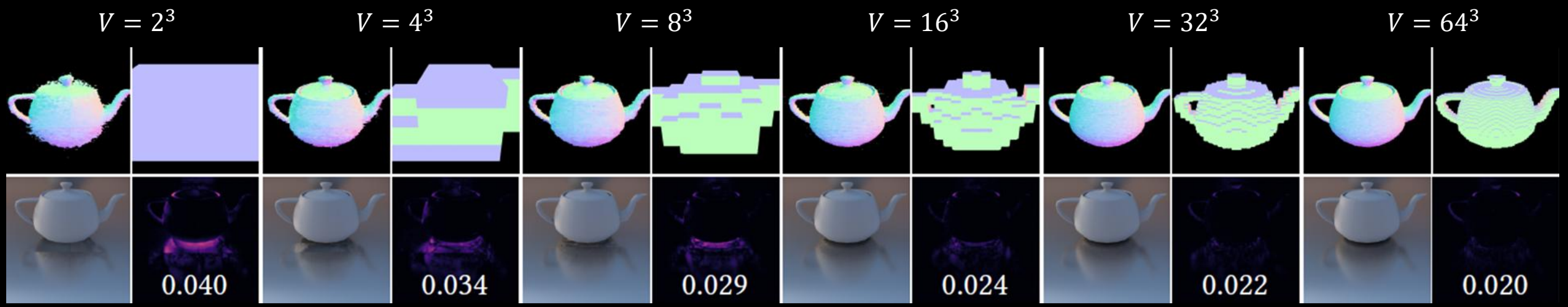
Parameter Study

- Parameters controlling the LSNIF accuracy

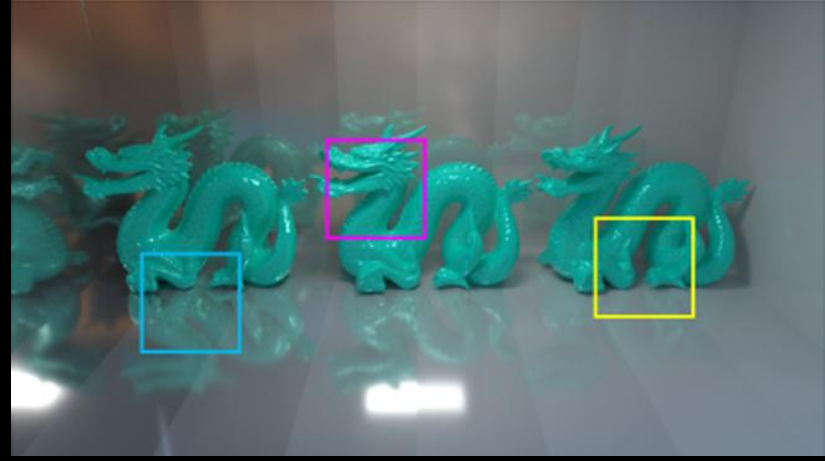
1. V : Voxel resolution
2. H: Upper bound for the number of intersection points,
3. M: Hash-map size in sparse hash grid

Parameter Study – Voxel Resolution

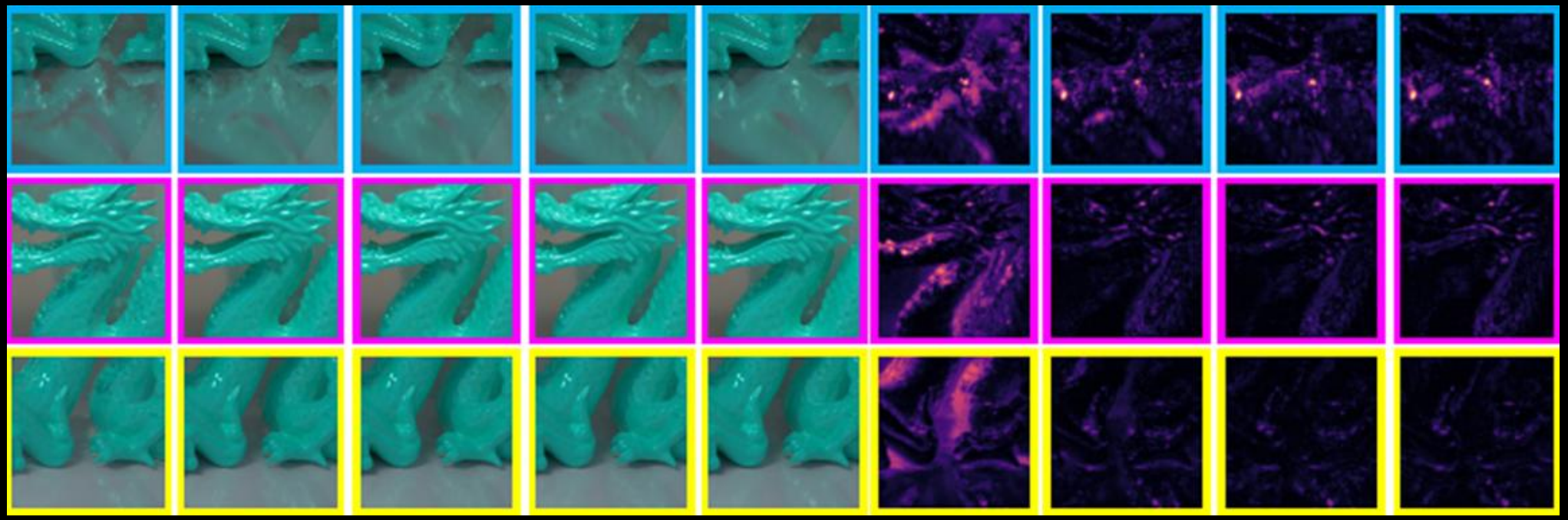
- Resolution to voxelize the local geometry
- With lower V, artifacts in reflection are visible
  - Reconstructed normal shows worse quality
- Higher V makes the voxel representation close to the geometry
  - $V = 32^3$ shows enough reconstruction quality



Parameter Study – Number of intersections

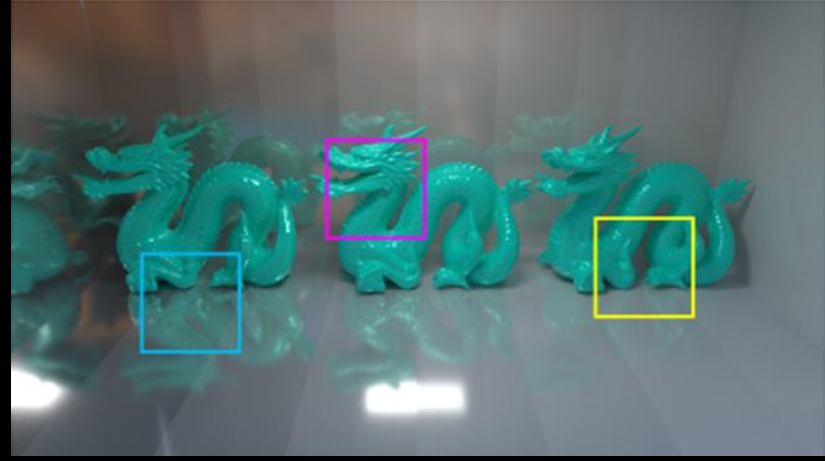


- Number of intersection points we collect for the NN inputs
- Smaller H fails to reconstruct the complex geometry
  - Insufficient capacity to capture all the his to voxels
- $H = 18$ shows good visual quality

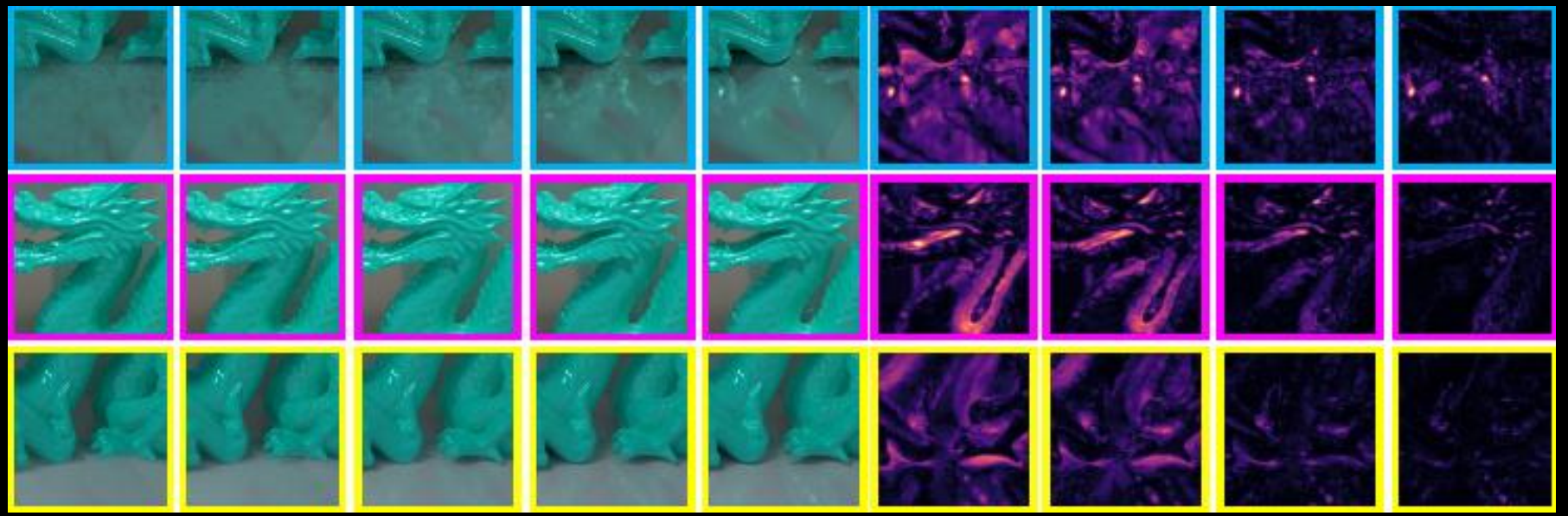


$H = 4$ $H = 8$ $H = 12$ $H = 18$ Ref. $H = 4$ 0.061 $H = 8$ 0.039 $H = 12$ 0.035 $H = 18$ 0.034

Parameter Study – Hash Map Size



- Hash map size in the sparse hash grid encoding
- Large part of the memory footprint of LSNIF
- Smaller M causes more hash collisions
  - Resulting in visual artifacts in shadows and reflections
- $M = 2^{17}$ shows good balance between quality and memory overhead



$M = 2^8$	$M = 2^{11}$	$M = 2^{14}$	$M = 2^{17}$	Ref.	$M = 2^8$	$M = 2^{11}$	$M = 2^{14}$	$M = 2^{17}$
3 KB	24 KB	192 KB	1.5 MB		0.085	0.061	0.047	0.034