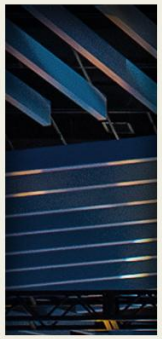
SIGGRAPH 2026
Los Angeles 19–23 JUL

The Premier Conference & Exhibition on
Computer Graphics & Interactive Techniques

Multi-feature Radiance Baking Neural Networks for Instant Volumetric Rendering

Jiaming Liang[1], Hongliang Yuan[2], Meng Gai[1], Guoping Wang[1], Sheng Li[1]

1: School of Computer Science, Peking University; 2: Xiaomi Company



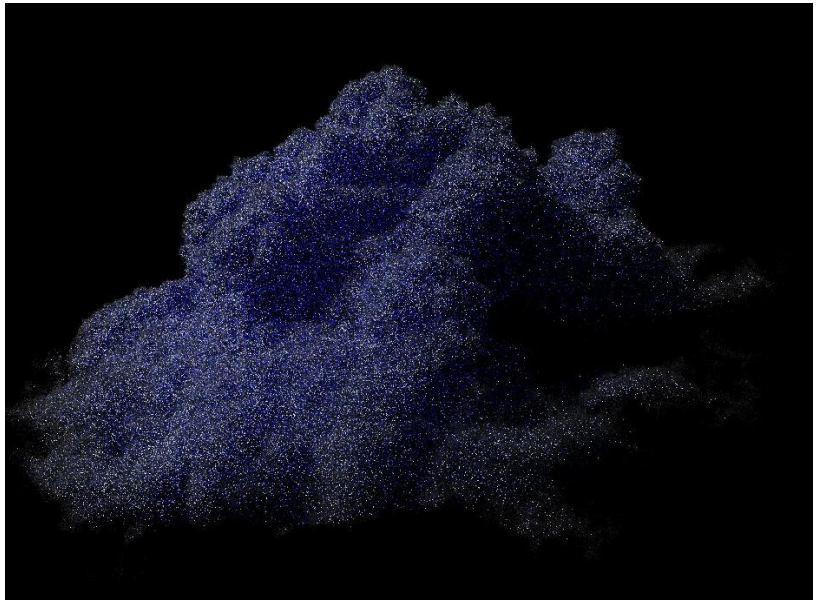


SIGGRAPH 2026
Los Angeles 19–23 JUL

Motivation

Motivation

- Traditionally, path tracing is used to achieve photorealistic volumetric rendering.
  - Core problem: too slow to converge in complex settings for real-time application.



25 ms (1 spp)

1 second

6 minutes

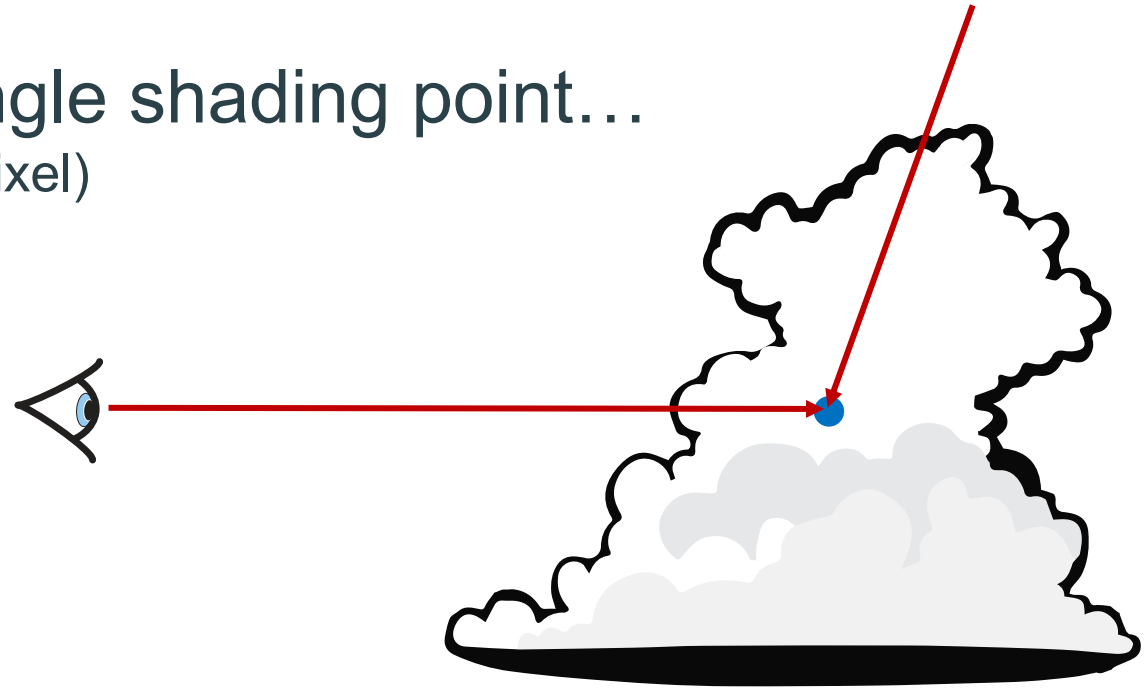
Motivation

- RPNN family tries to accelerate this process by predicting radiance directly.
  1. Construct stencil and sample features from volume;
  2. Fuse and map these features to scattering radiance;
  3. Add direct illumination and accumulate them to get radiance received from camera.
- RPNN: minutes to converge;
- MRPNN: 40 FPS per spp, allows phase and albedo change;
  - **But still too costly!**

Motivation

- Question: where do costs of RPNN family come from?

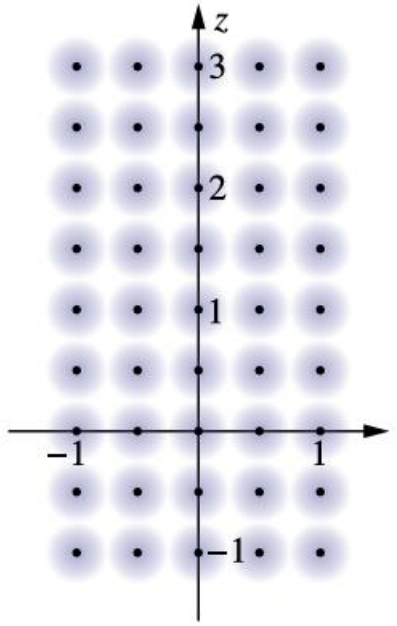
Given a single shading point…
(e.g. for each pixel)



$$\gamma = \operatorname{acos}(\omega \cdot \omega_s) = \langle \omega, \omega_s \rangle$$

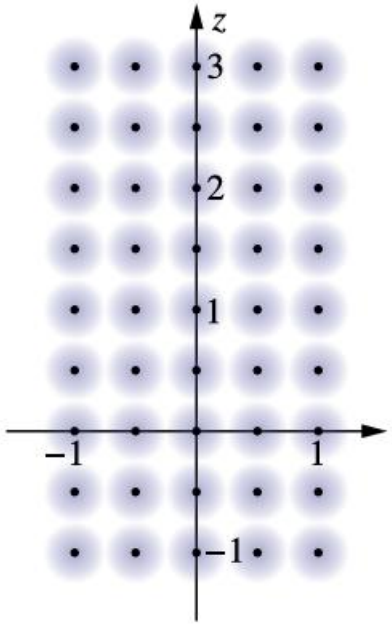
Figure courtesy of RPNN [Kallweit et.al. 2017].

RPNN Pattern:
$5 \times 5 \times 9,\ 10$ layers

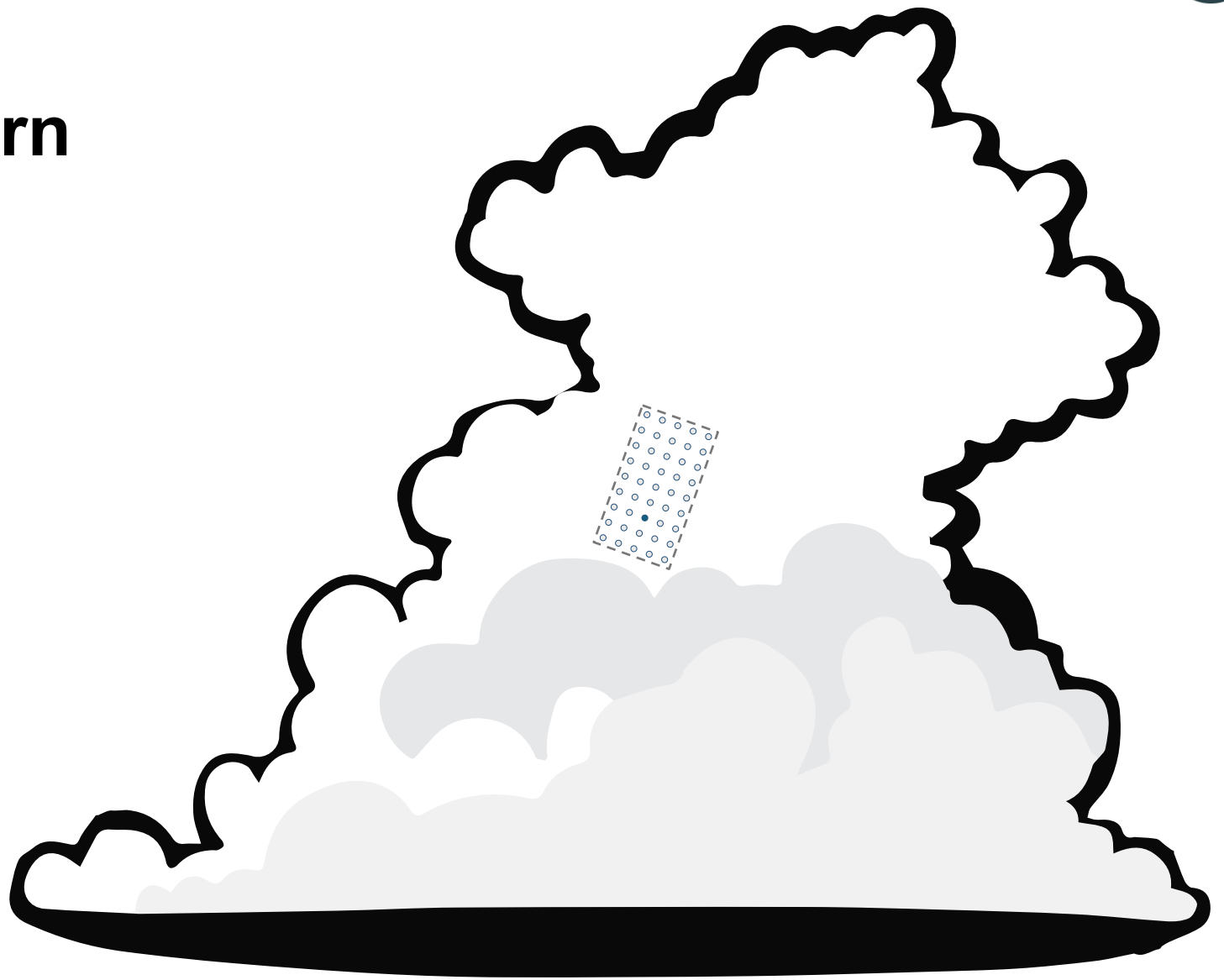


Σ^0

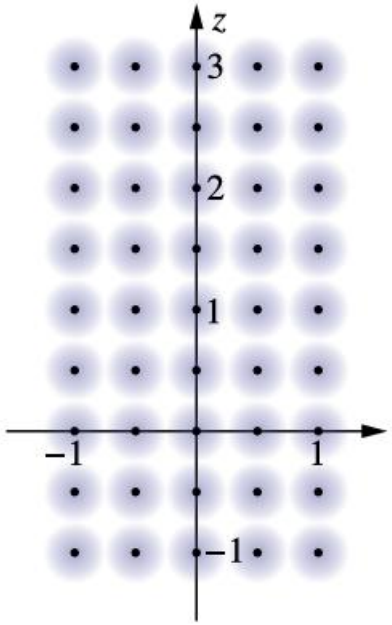
RPNN Pattern



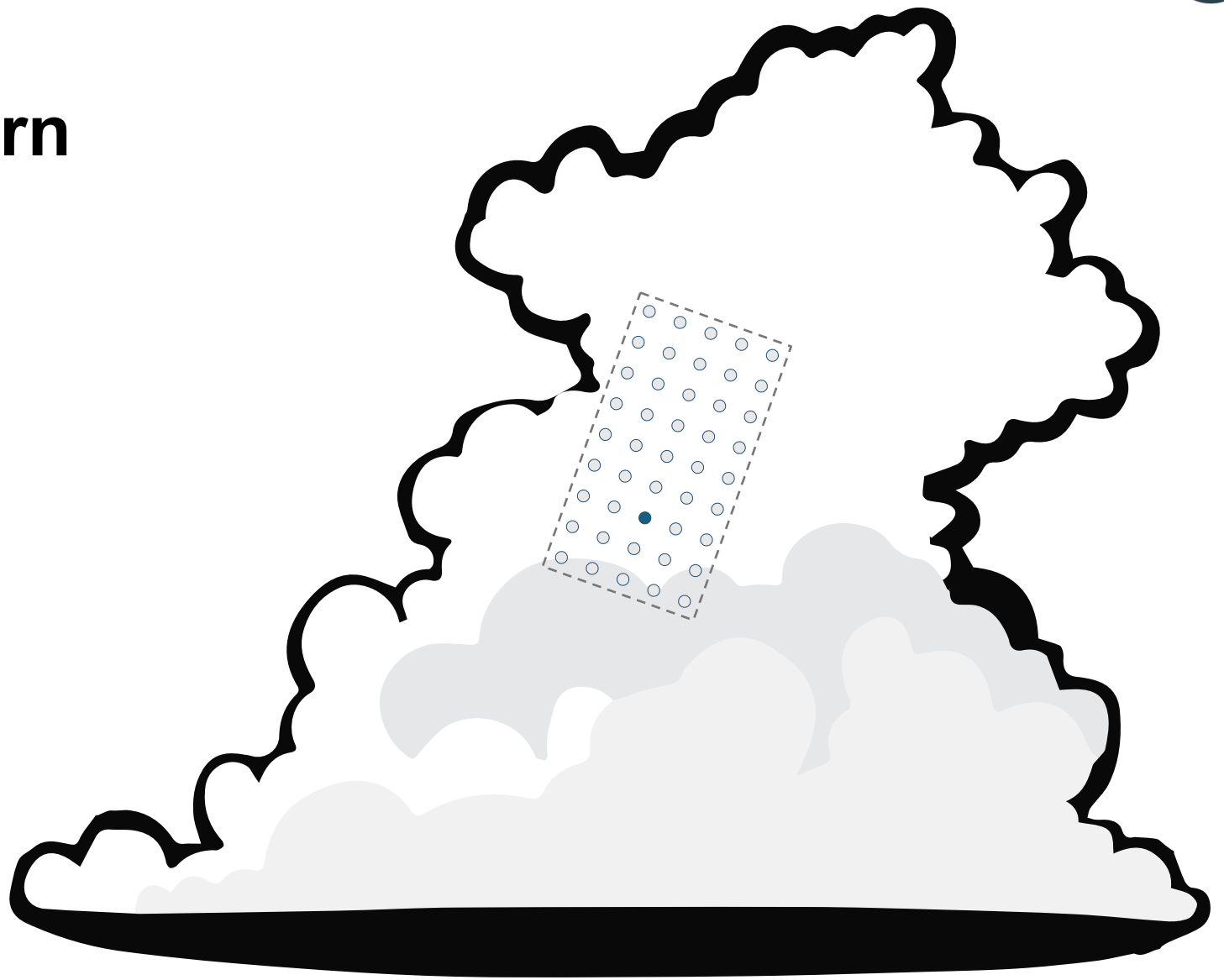
Σ^1



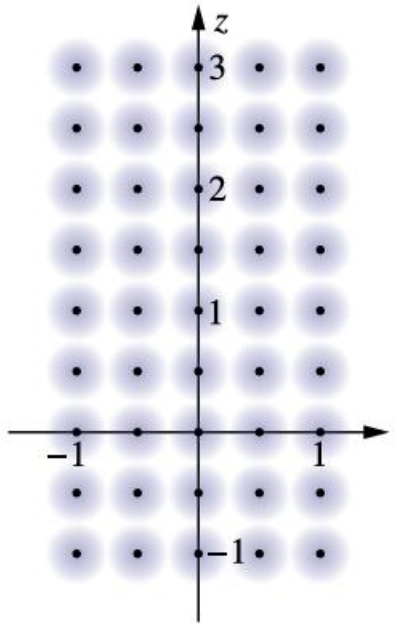
RPNN Pattern



Σ^2

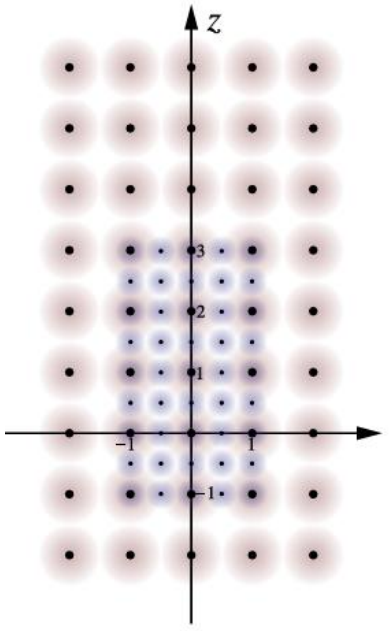


RPNN Pattern

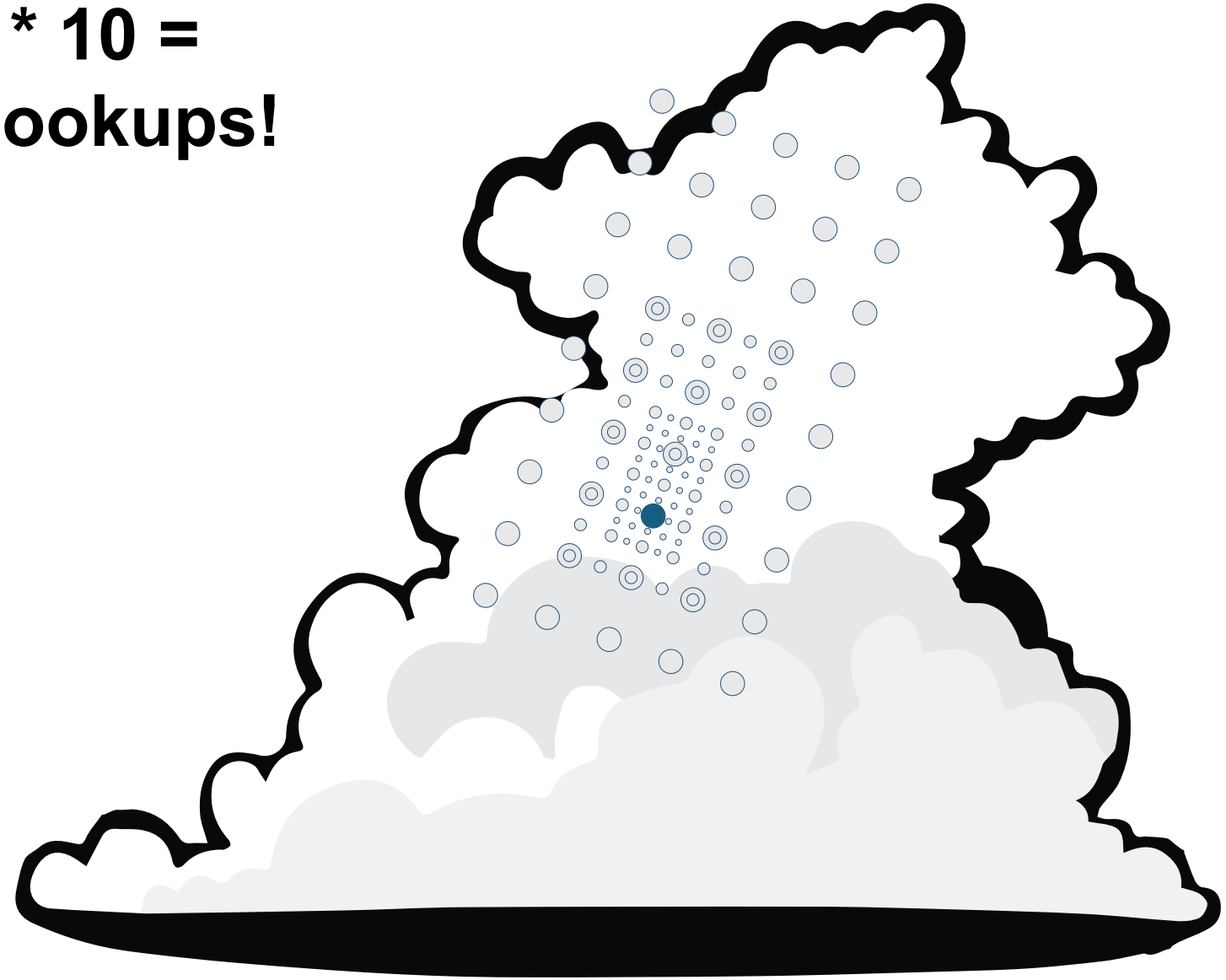


Σ^3

Totally 225 * 10 = 2250 texture lookups!



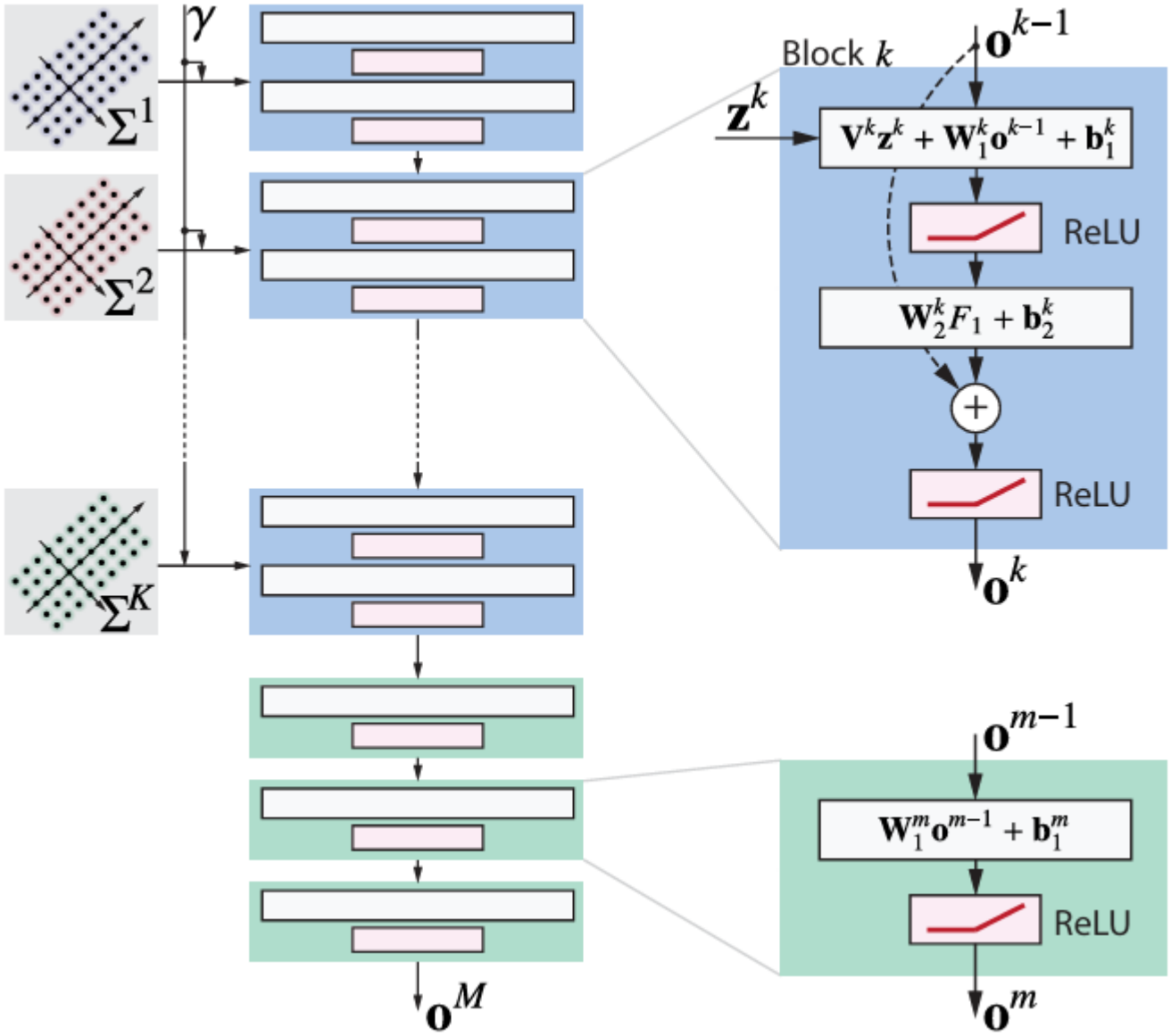
Σ



RPNN Architecture:
$10 * 2 + 3$ FC layers
1.3M Parameters

Too much lookup and inference cost for a single spp… 😵

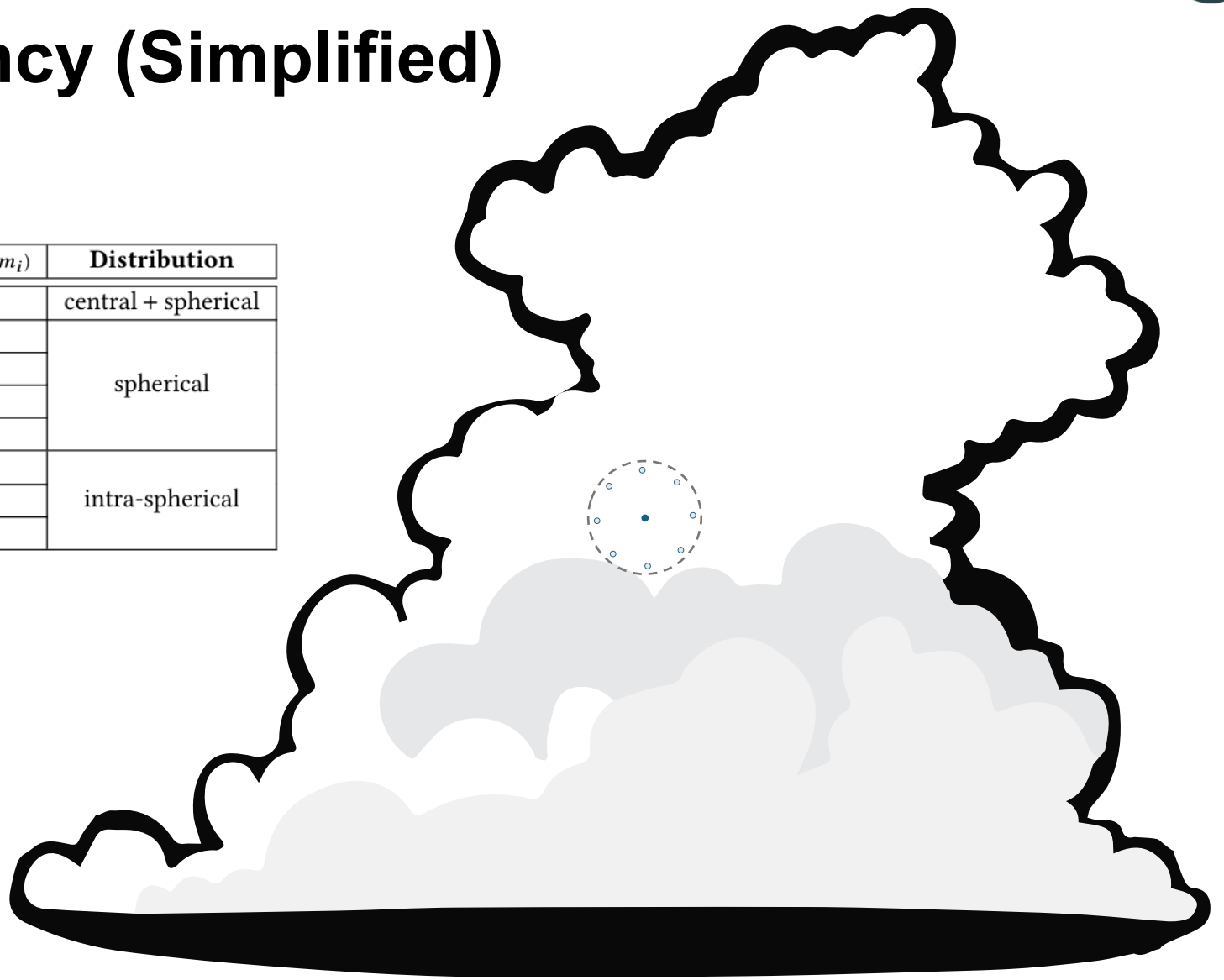
Core reason: difficult to learn mapping from smooth density to high-frequency radiance.



MRPNN Pattern:
Low frequency (Simplified)

Layer (i)	Num. Points (N_i)	Mip-level (m_i)	Distribution
1	8	0	central + spherical
2		1	spherical
3	16	2	
4		3	
5		4	
6	32	5	intra-spherical
7		6	
8		7	

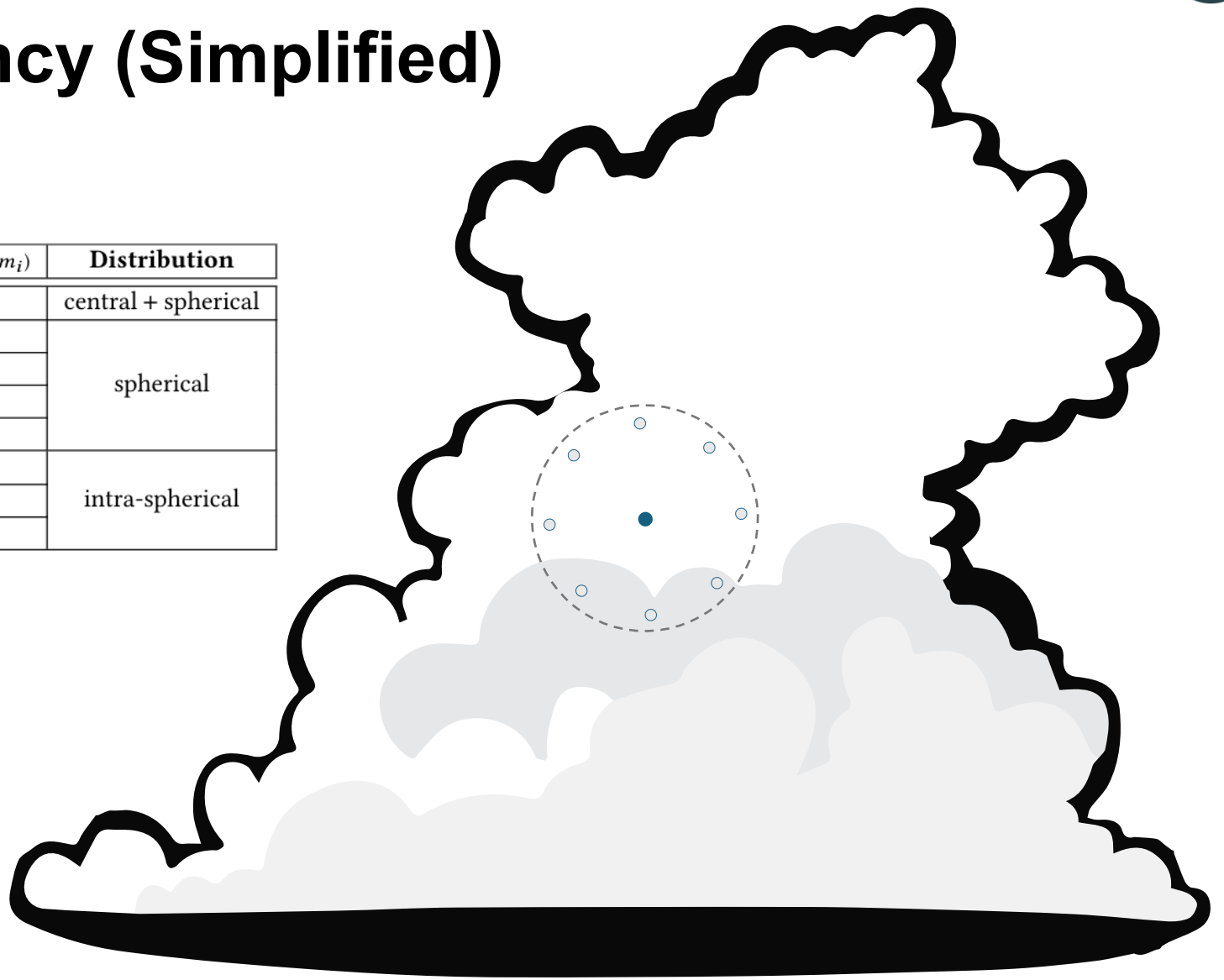
Σ^0



MRPNN Pattern:
Low frequency (Simplified)

Layer (i)	Num. Points (N_i)	Mip-level (m_i)	Distribution
1	8	0	central + spherical
2		1	spherical
3	16	2	
4		3	
5		4	
6	32	5	intra-spherical
7		6	
8		7	

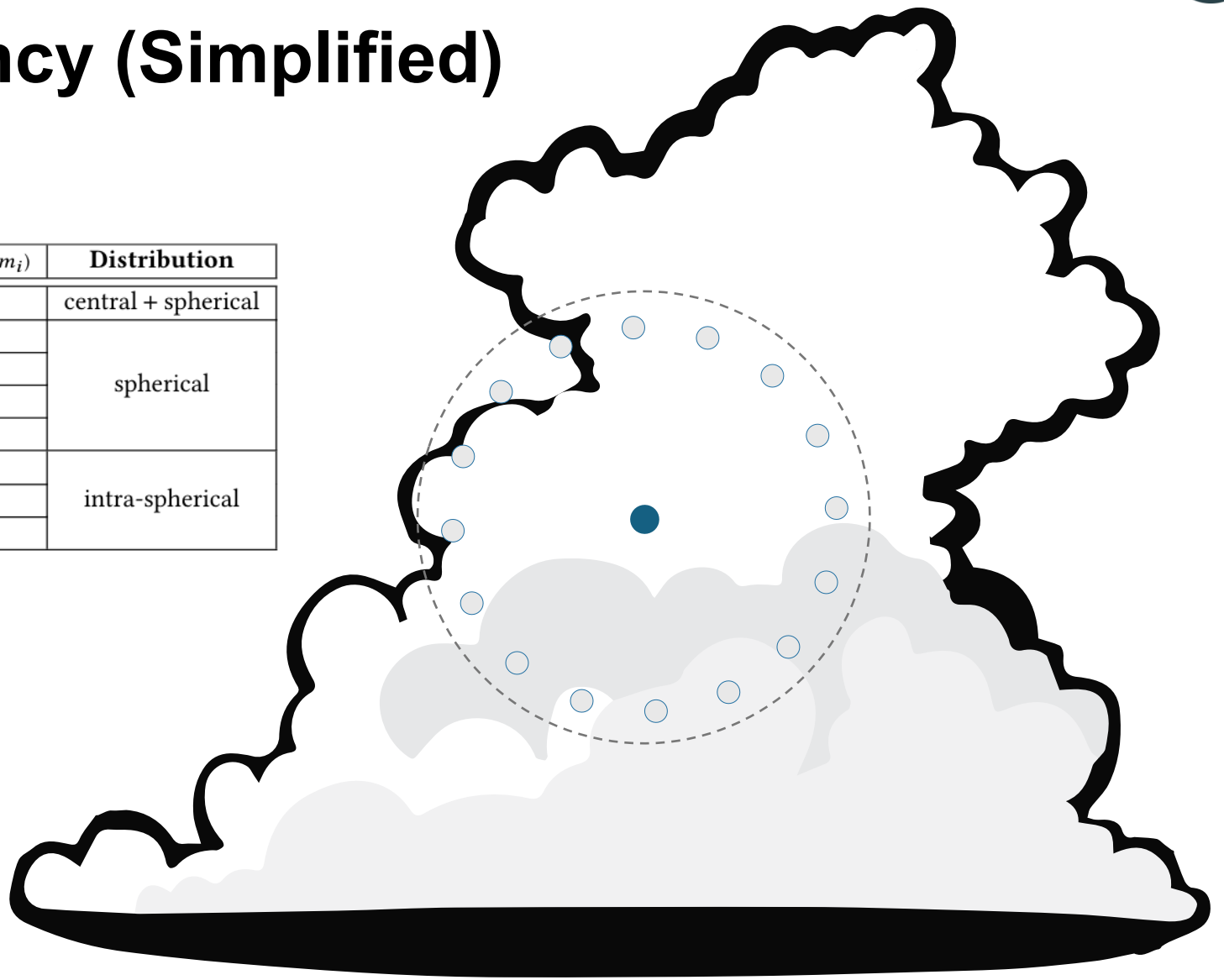
Σ^1



MRPNN Pattern:
Low frequency (Simplified)

Layer (i)	Num. Points (N_i)	Mip-level (m_i)	Distribution
1	8	0	central + spherical
2		1	spherical
3	16	2	
4		3	
5		4	
6	32	5	intra-spherical
7		6	
8		7	

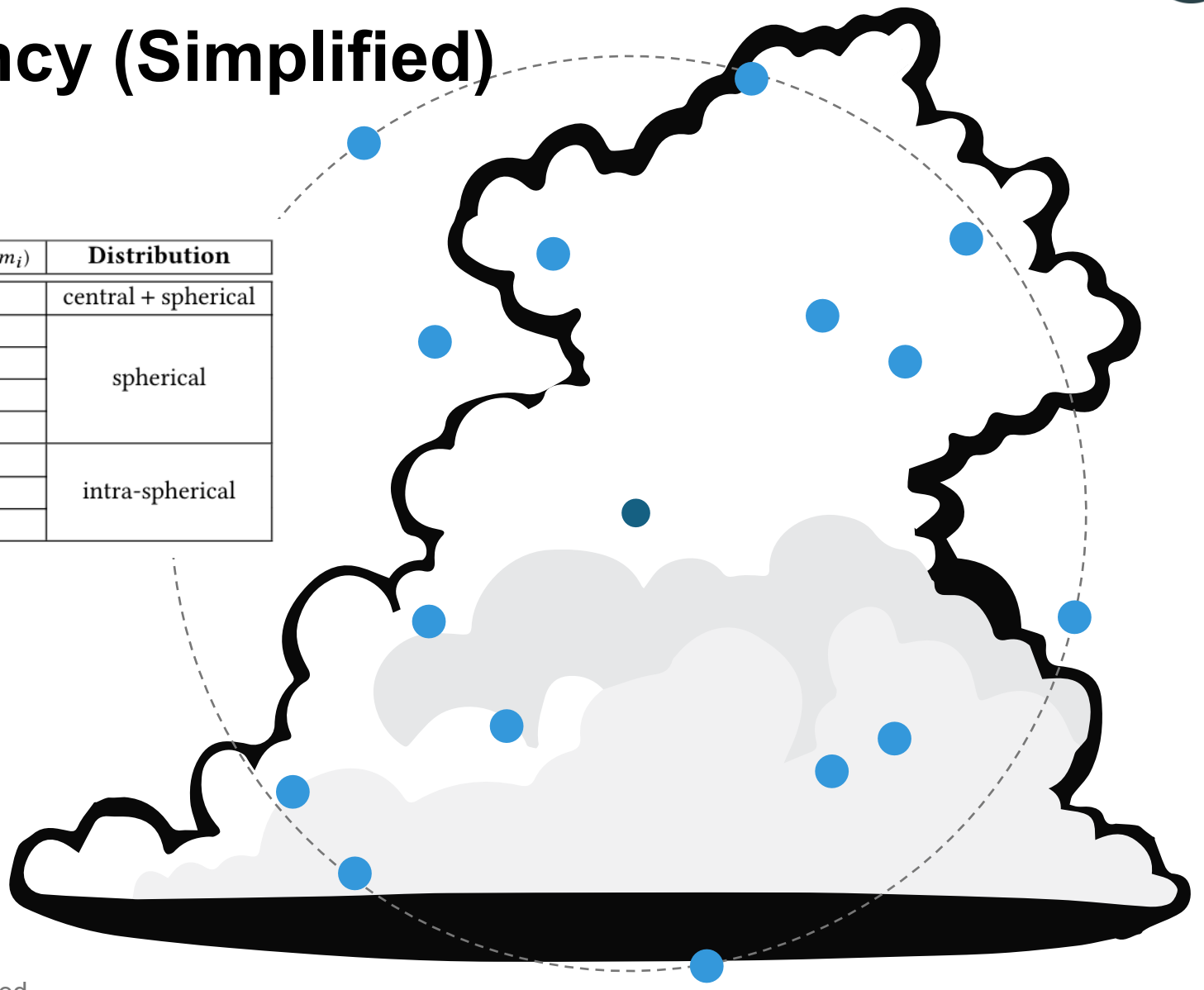
Σ^2



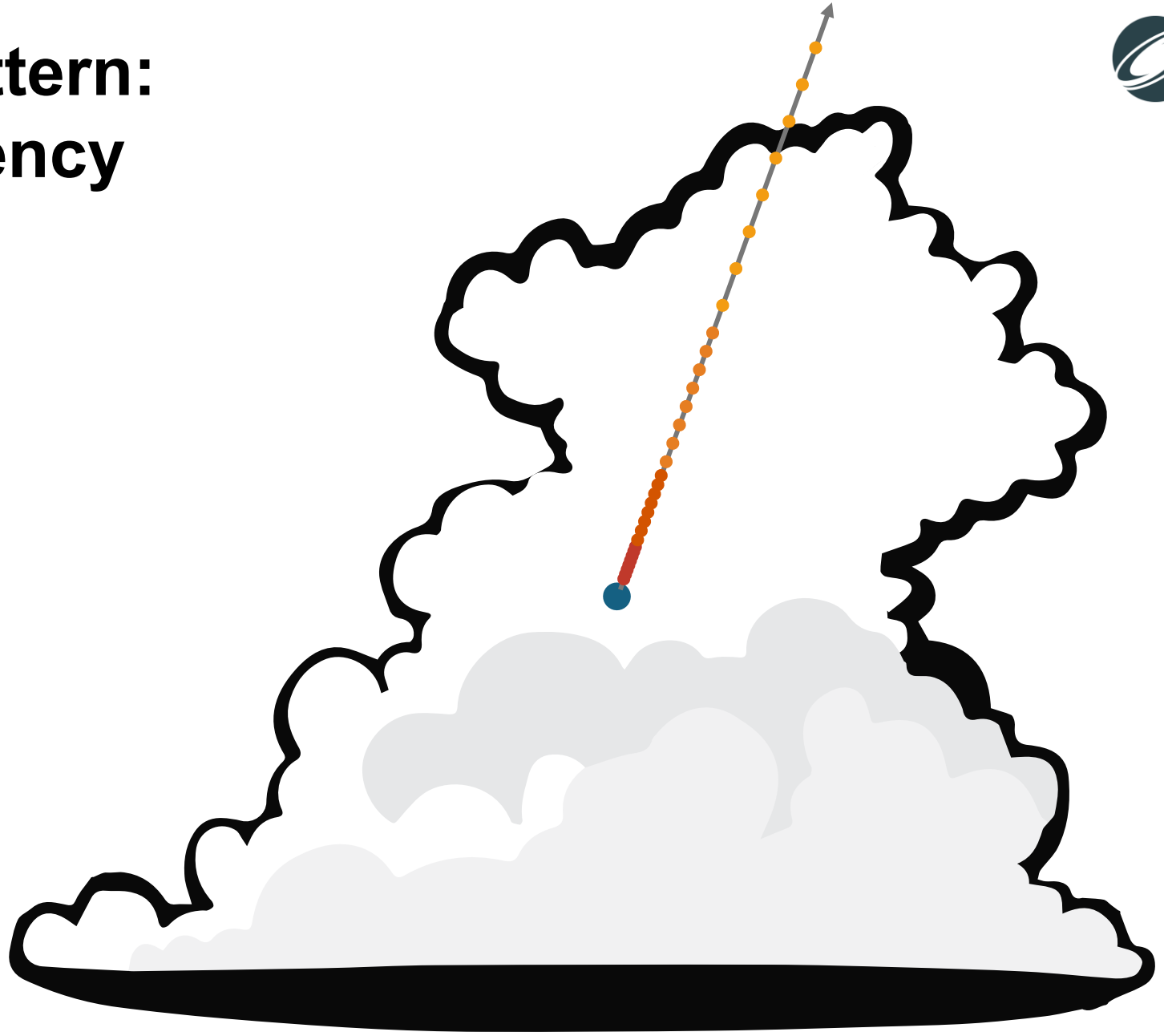
MRPNN Pattern:
Low frequency (Simplified)

Layer (i)	Num. Points (N_i)	Mip-level (m_i)	Distribution
1	8	0	central + spherical
2		1	spherical
3	16	2	
4		3	
5		4	
6	32	5	intra-spherical
7		6	
8		7	

Σ^5



MRPNN Pattern:
High frequency



MRPNN Pattern: The Real Overall

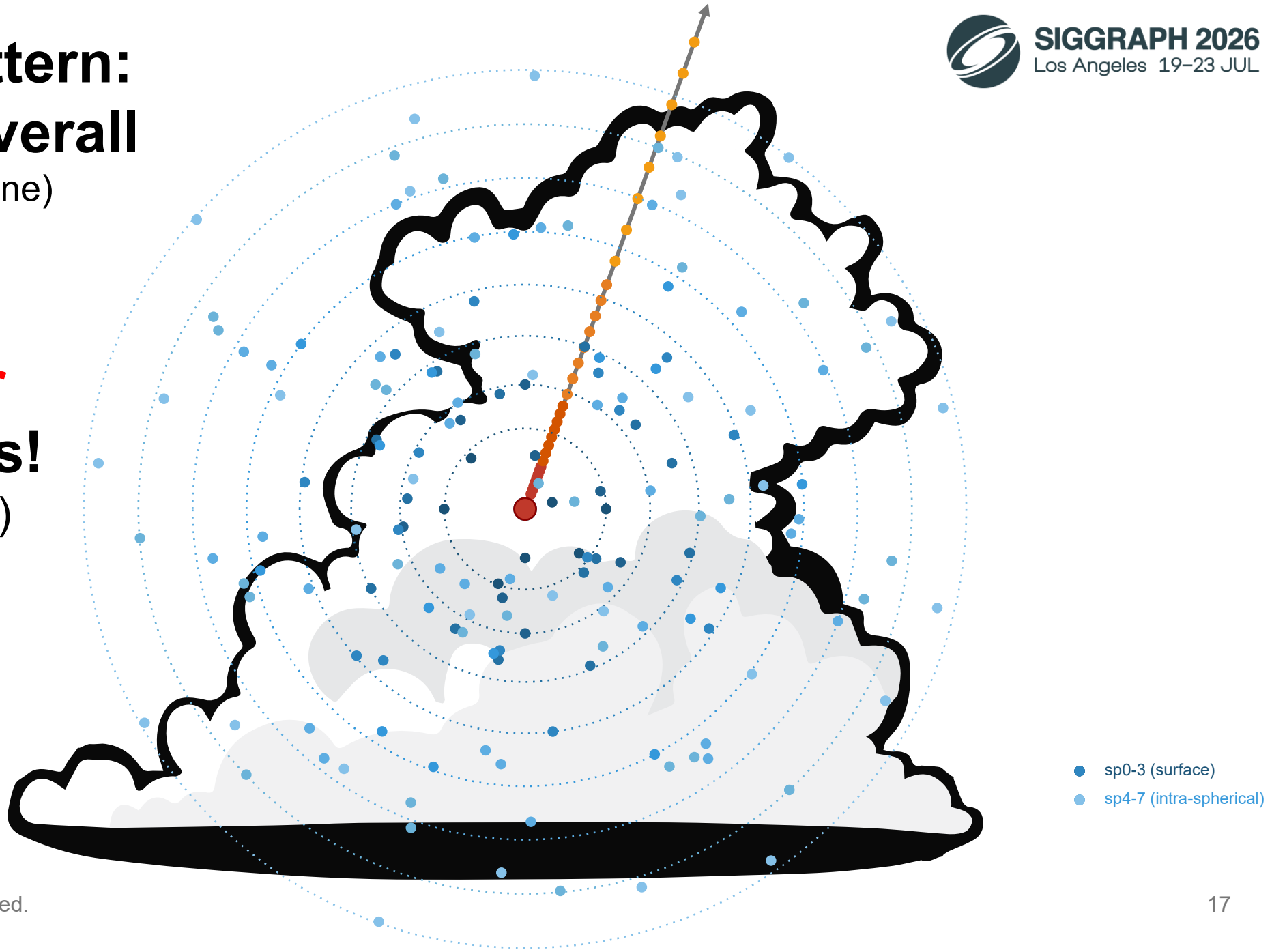
(projected on xy plane)

Totally

~800 irregular

texture lookups!

(with phase support)



MRPNN Architecture:

57 FC layers (with other operations like pooling)

50K Parameters but complex inference process

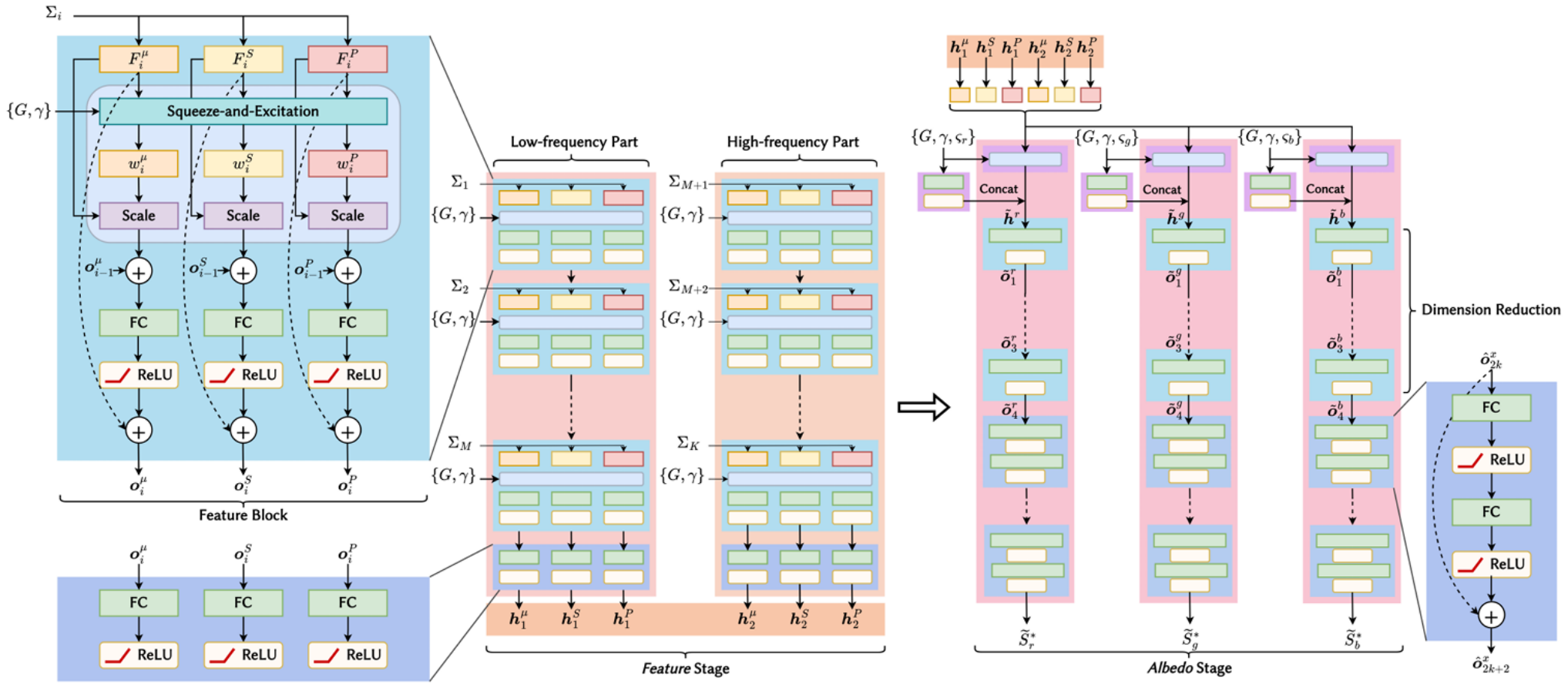
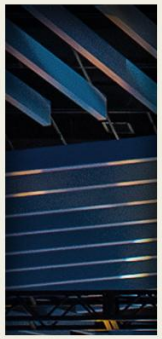


Figure courtesy of MRPNN [Hu et.al. 2023].

Motivation

- Apparently, MRPNN is still dragged by these two factors:
  1. Too many irregular texture lookups, to describe surrounding environment;
  2. Too complex and deep network, to fuse all features.
- Our solution: compact them by a baked latent-space representation!
  1. Single sampling point ⇒ a scope in the original space;
  2. Easier mapping ⇒ even more lightweight structure;
  3. Possible to bake more complex settings such as albedo grid and environment light.





SIGGRAPH 2026
Los Angeles 19–23 JUL

Check our paper for details!

Method & Design

Neural Baking Design

- So now there exist three core problems:
  1. What to sample?
     (Design of latent-space representation)
  2. How to sample?
     (Design of sampling pattern)
  3. How to map?
     (Design of neural decoder)

- Observe VRTE:

Estimation goal $S(p,\omega)$

$$L_s(p,\omega) = \sigma_a(p,\omega)L_e(p,\omega) + \sigma_s(p,\omega)\int_{S^2}\phi(p,\omega_i,\omega)L(p,\omega_i)\mathrm{d}\omega_i,$$

$$L(p,\omega) = \int_0^\infty T_r(p'\rightarrow p)L_s(p',\omega)\mathrm{d}t$$

  - RHS vs. LHS
  - Can we find a representation to reduce RHS cost?
  - **By spherical harmonics**.

Neural Baking Design

- Key observation: practical phase functions are all zonal, i.e. $\phi(\omega_i \cdot \omega)$.
  - Zonal convolution in integral ⇔
    Weighted sum in spherical harmonics!

THEOREM 3.1. *Let radiance field SH decomposition be* $L(p,\omega) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} L_l^m(p)Y_l^m(\omega)$, *the in-scattering field can be expressed as:*

$$S(p,\omega) = \sum_{l=0}^{\infty} \eta_l(p) \sum_{m=-l}^{l} L_l^m(p)Y_l^m(\omega)$$

where $\eta_l(p) = \frac{4\pi f_l(p)}{2l+1}$ *and* $f_l(p)$ *is the* l*-th order coefficients of Legendre polynomials* P_l *decomposed from phase function* ϕ *at position* p, *and* Y_l^m *is the SH of degree* l *and order* m.

- But still two problems:
  1. It assumes decomposition L_l^m is known, which is impossible in dynamic conditions.
  2. The series is infinite so there exists systematic bias when used with truncation.

Neural Baking Design

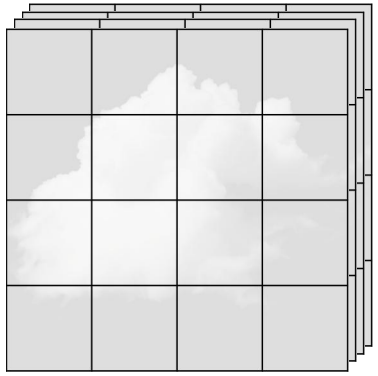
$$S(p,\omega) = \sum_{l=0}^{\infty} \eta_l(p) \sum_{m=-l}^{l} L_l^m(p) Y_l^m(\omega)$$

- Solution: represent L_l^m with latent-space $\mathcal{L}_l^m$ and decode it.
- Intuitively: $S \approx \sum_{l=0}^{K} \eta_l \sum_m \mathcal{S}(\mathcal{L}_l^m) Y_l^m$, where $\mathcal{S}$ is decoder;
  - However, we still need to do inference for many times.
  - So instead, we assume that this transformation can also be applied in latent space!
- Finally: $\tilde{S} = \sum_{l=0}^{K} \eta_l \sum_m \mathcal{L}_l^m Y_l^m$, $S = \mathcal{S}(\tilde{S})$.

Neural Baking Design

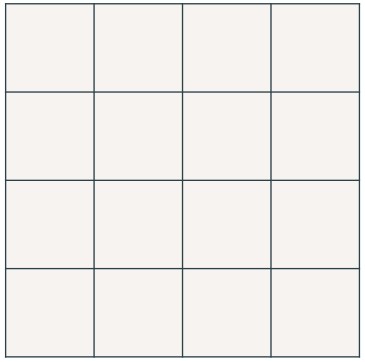
- Introducing dynamic conditions on view / light direction, phase and albedo, our factorized design is:

3D



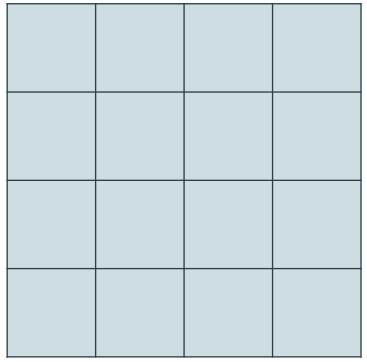
SH Grid $\mathcal{L}_l^m(p)$

2D



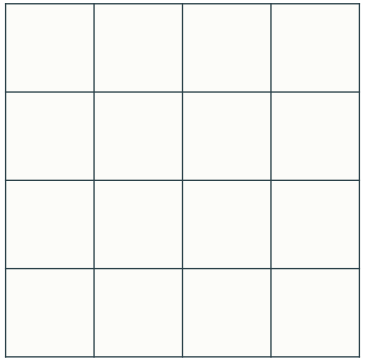
View Grid $\mathcal{V}(\omega)$

2D



Phase Grid $\mathcal{G}(g, \cos\gamma)$

2D



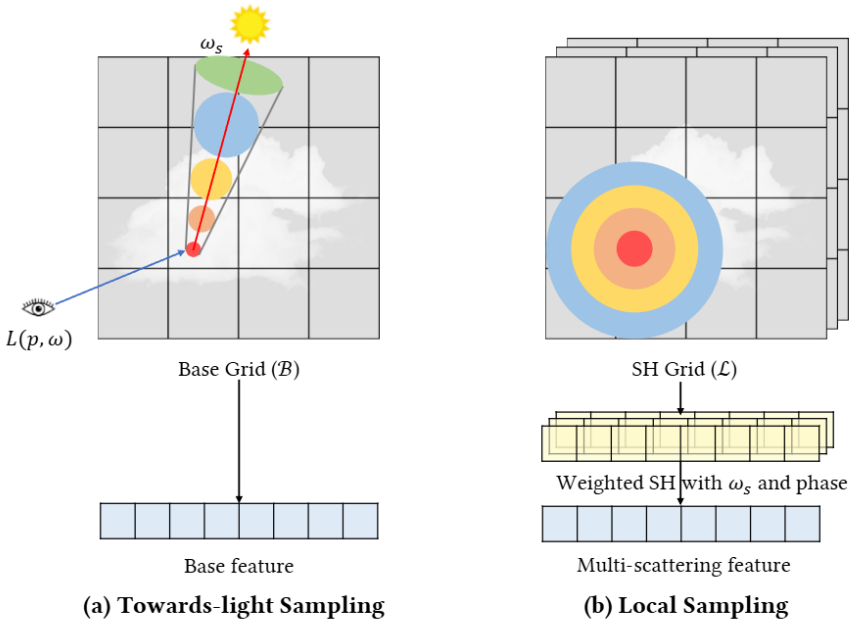
Albedo Grid $\mathcal{A}(\alpha, \cos\gamma)$

Sampling Pattern Design

- Dual-pattern sampling:
  - Low-frequency: sample locally;
  - High-frequency: sample towards light.
  - Since we sample in latent space, we just sample **one point** in every LoD.
  - Naïvely, we can just use these two patterns on the SH grid.
    - However, every SH grid sample needs $(K+1)^2$ underlying samples;

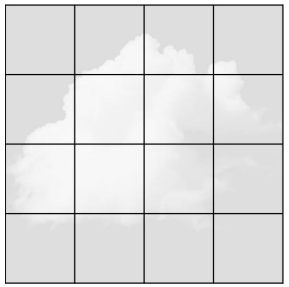
Sampling Pattern Design

- Alternatively:
  - Low-frequency pattern on the SH grid;
  - High-frequency pattern on a normal base grid plus transmittance map.
    - $(K+1)^2$ grids to 2 grids!

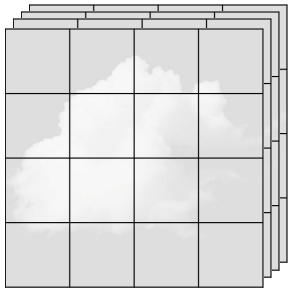


Sampling Pattern Design

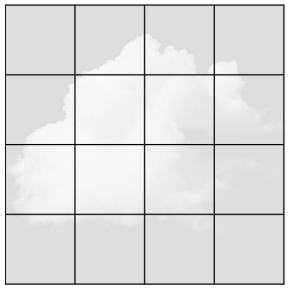
- So the final baking design is:



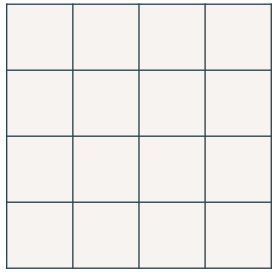
Base Grid
$\mathcal{B}(p)$



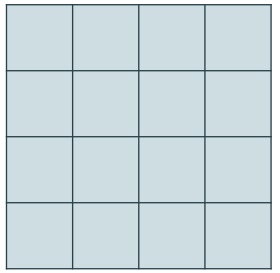
SH Grid
$\mathcal{L}_l^m(p)$



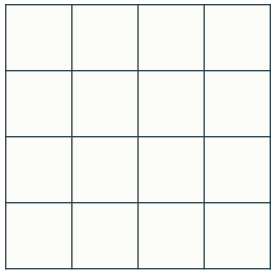
Transmittance
Map $\mathcal{T}(p)$



View Grid
$\mathcal{V}(\omega)$



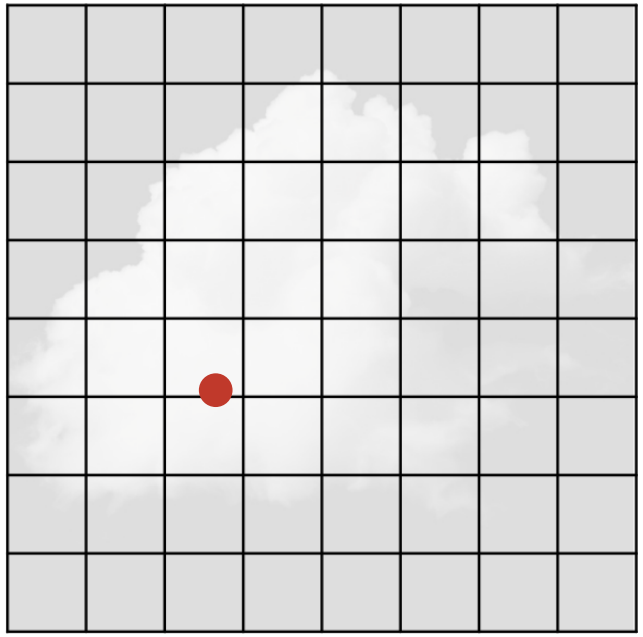
Phase Grid
$\mathcal{G}(g, \cos\gamma)$



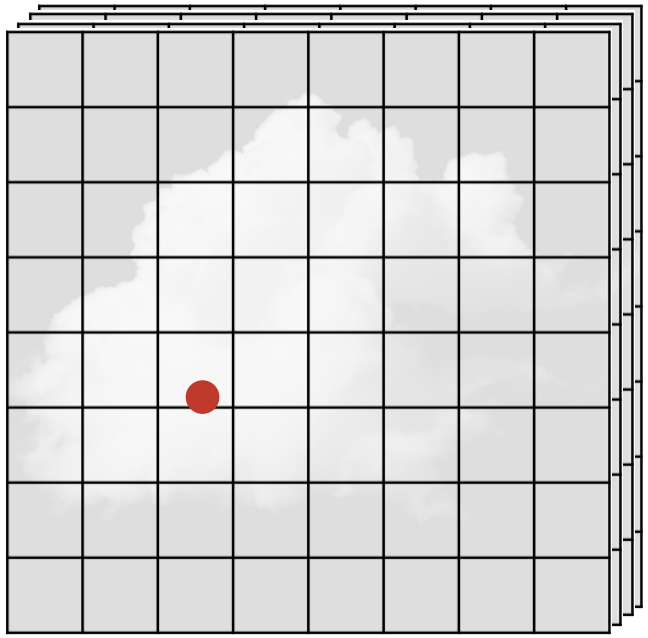
Albedo Grid
$\mathcal{A}(\alpha, \cos\gamma)$

- Then, we can extract features from each layer;
  - Take four LoDs and steps as example;

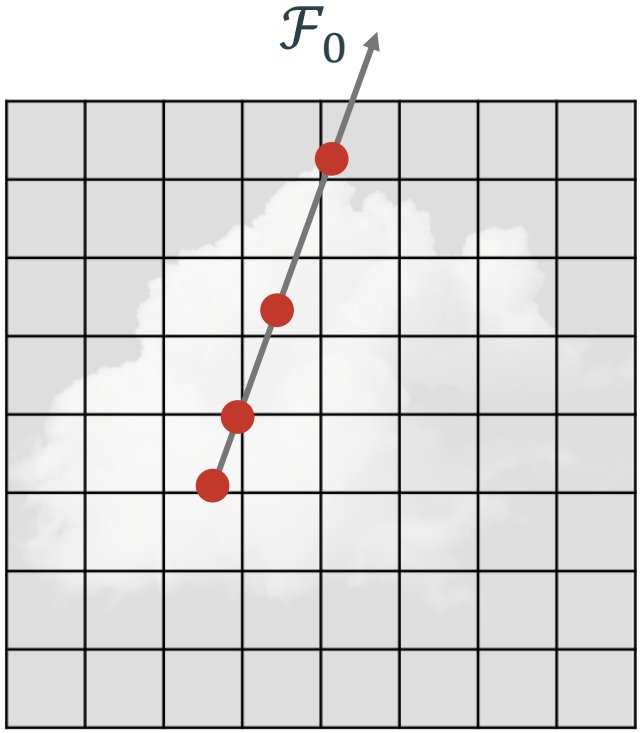
Sampling Pattern Design



Base Grid $\mathcal{B}(p_0)$



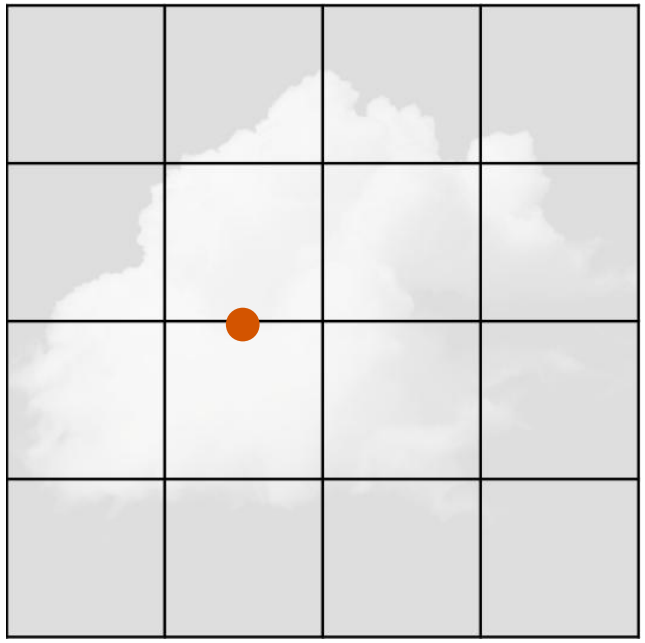
SH Grid $\mathcal{L}_l^m(p_0)$



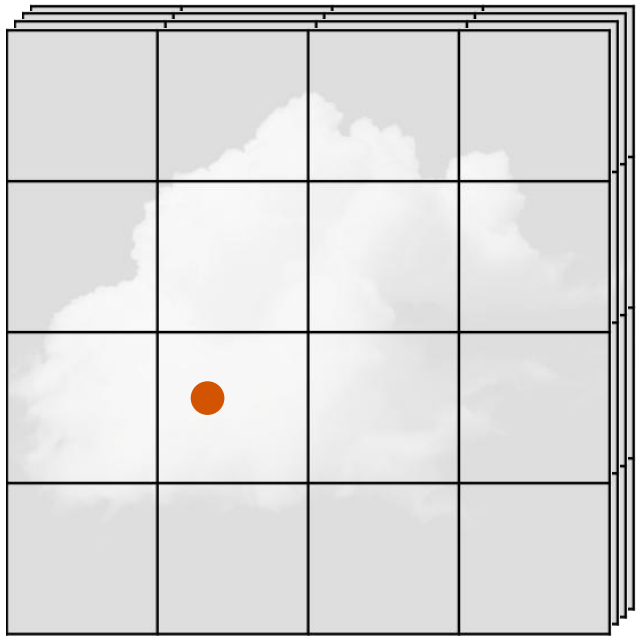
Transmittance Map $\mathcal{T}(p_j)$

Transmittance map doesn't lie in latent space so we sample all points in every layer.

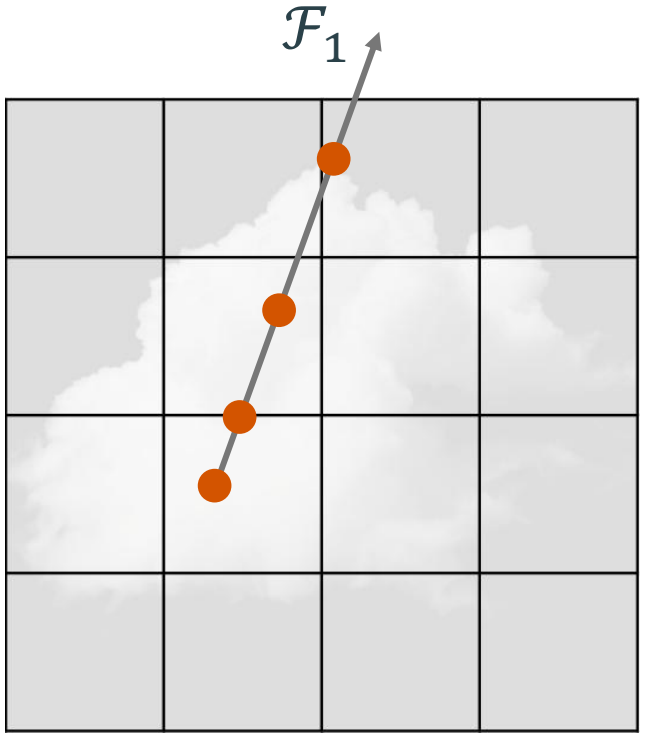
Sampling Pattern Design



Base Grid $\mathcal{B}(p_1)$



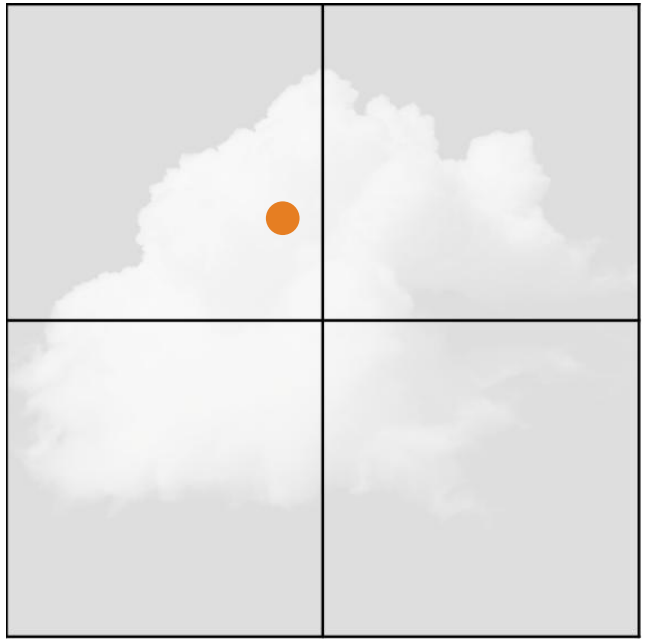
SH Grid $\mathcal{L}_l^m(p_0)$



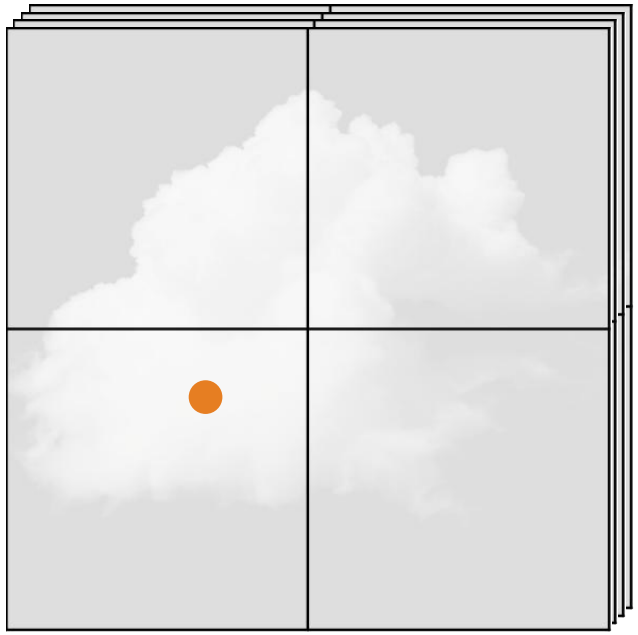
Transmittance Map $\mathcal{T}(p_j)$

Raise LoD to higher level, and advance to next sampling point.

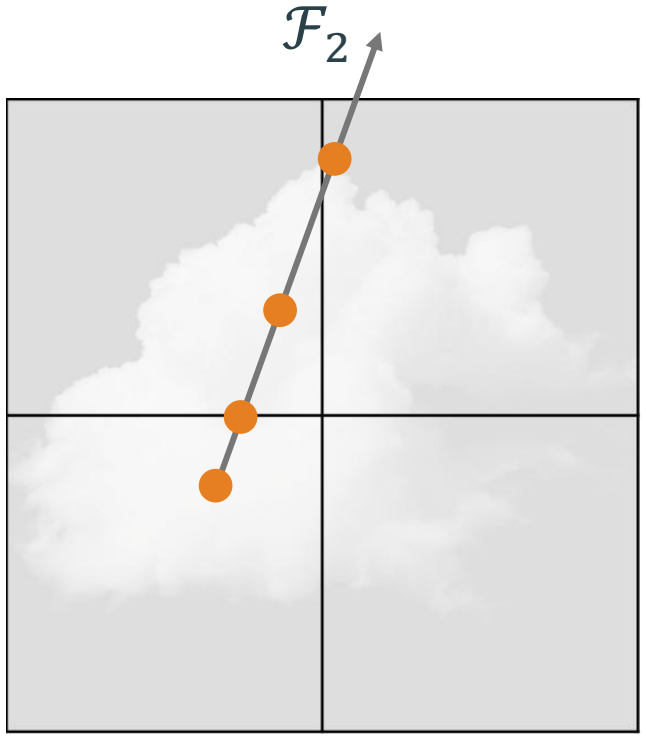
Sampling Pattern Design



Base Grid $\mathcal{B}(p_2)$

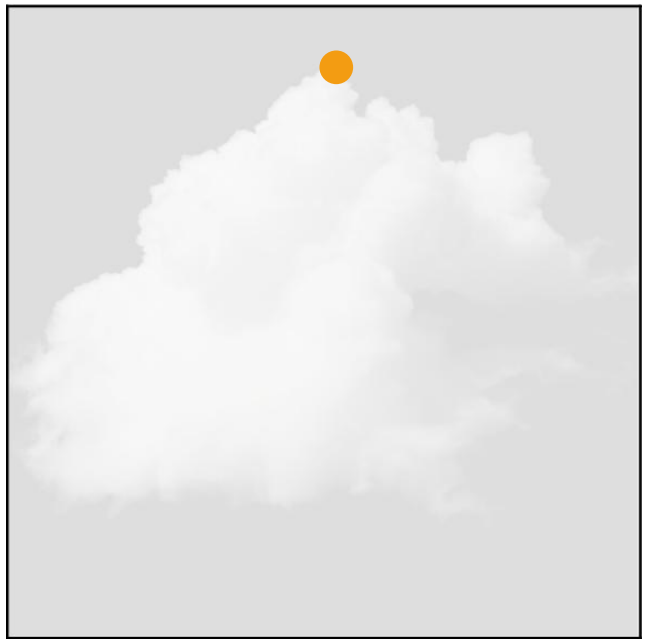


SH Grid $\mathcal{L}_l^m(p_0)$

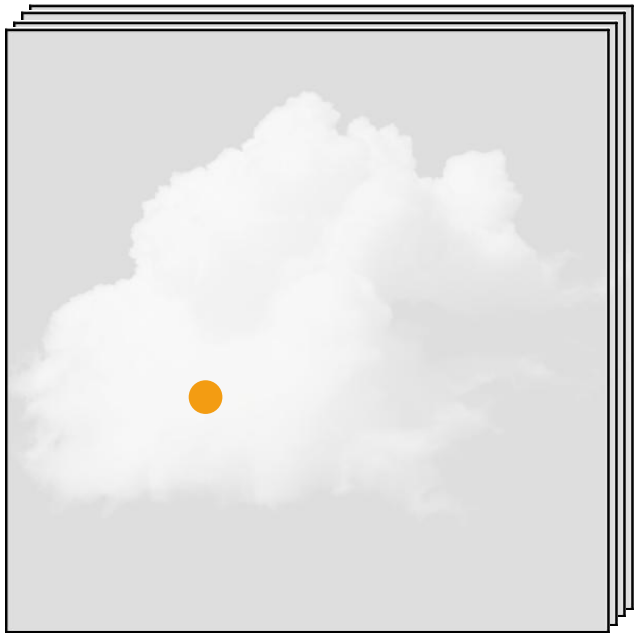


Transmittance Map $\mathcal{T}(p_j)$

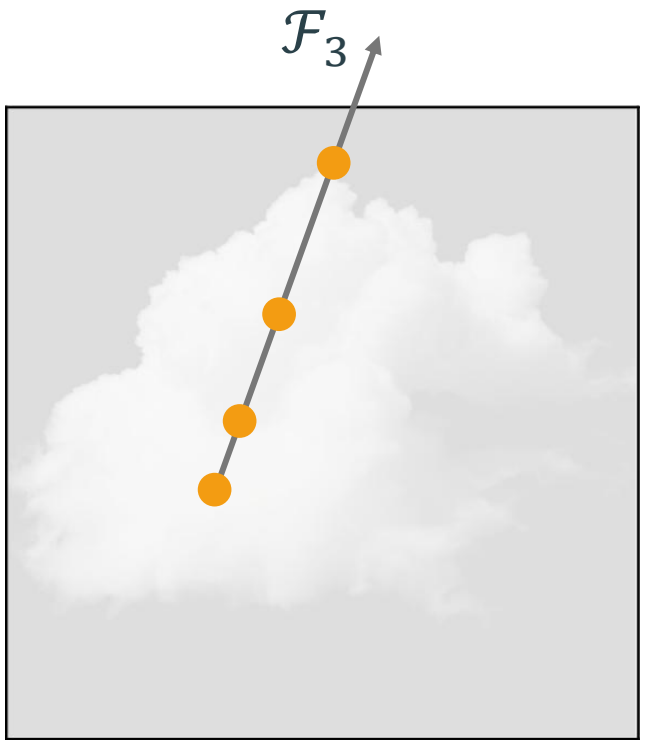
Sampling Pattern Design



Base Grid $\mathcal{B}(p_3)$



SH Grid $\mathcal{L}_l^m(p_0)$

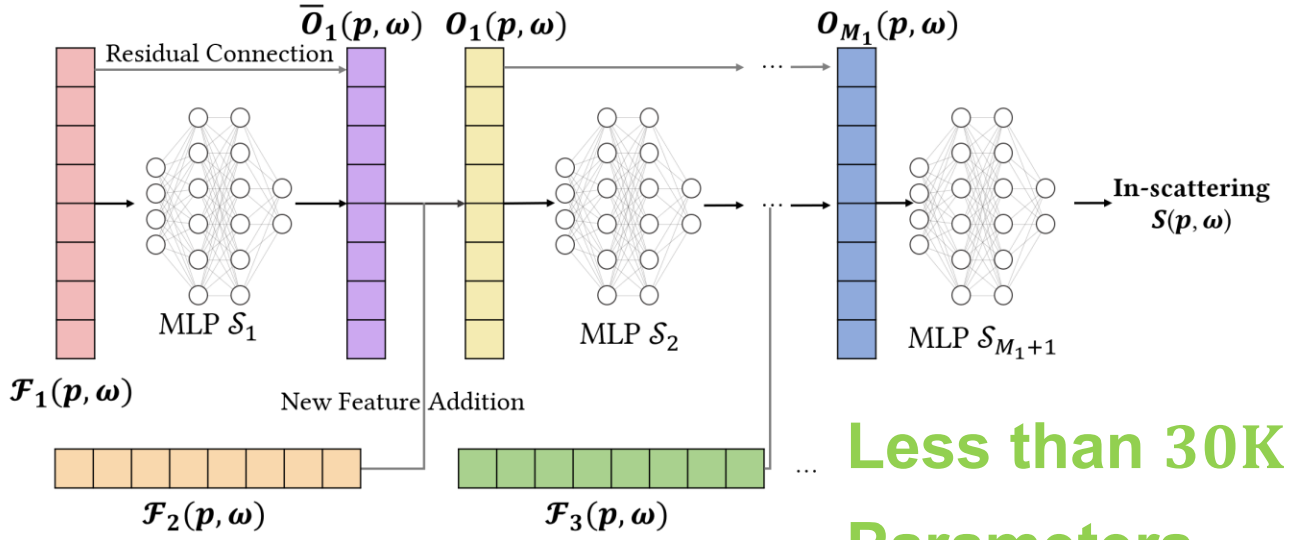


Transmittance Map $\mathcal{T}(p_3)$

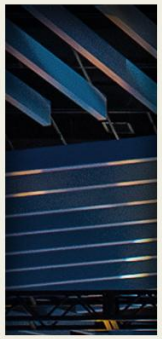
Very simple lookup pattern!

Decoder Design

- We aggressively simplify the network structure.
  - Core of RPNN: transform low-level features, fuse them with high-level features.
  - So our design is:
    1. Transform by Residual-Connected MLP;
    2. Fuse by addition;
    3. After fusing all features, map to radiance with a MLP.



Less than 30K Parameters





The Premier Conference & Exhibition on
Computer Graphics & Interactive Techniques

SIGGRAPH 2026
Los Angeles 19–23 JUL

Check our paper for details!

Experiments

Experiments

- Performance test:

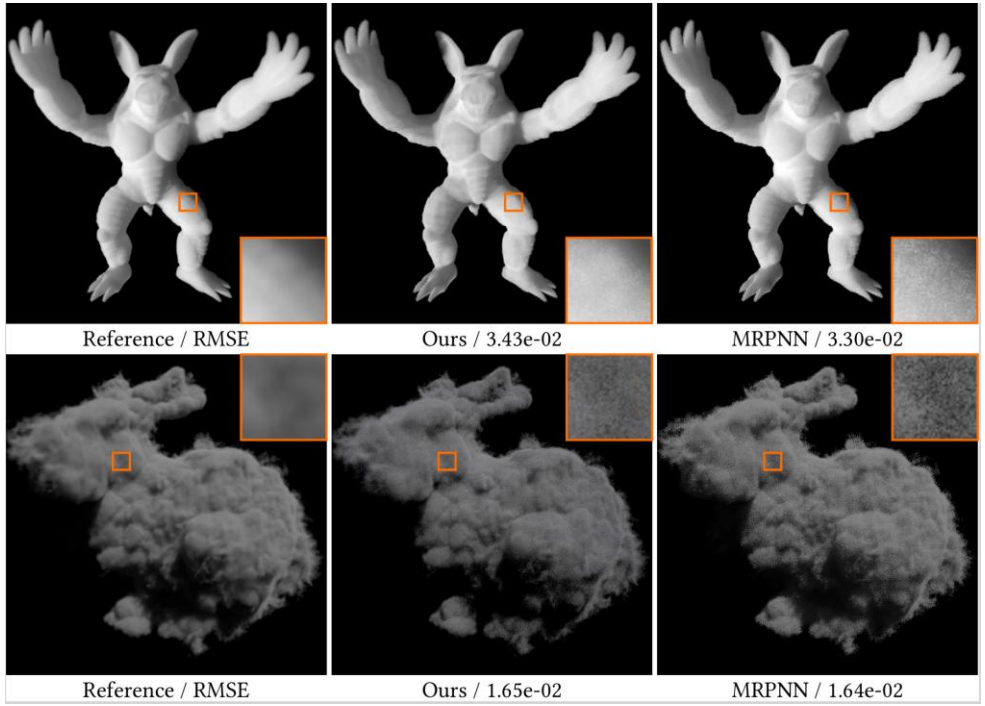
  (Disney cloud)

	Delta tracking	Sampling / Accessed elements	MLP inference	Post-process	Final FPS
MRPNN (single)	1.09	4.12 / 766	13.9	< 0.500	51.1
MRPNN (full)			19.0		40.5
Ours (single)		**2.55** / 432	**0.653**		207 (**4.05**×)
Ours (full)			**0.753**		203 (**5.01**×)
SH for base grid (full)		3.12 / 688			182

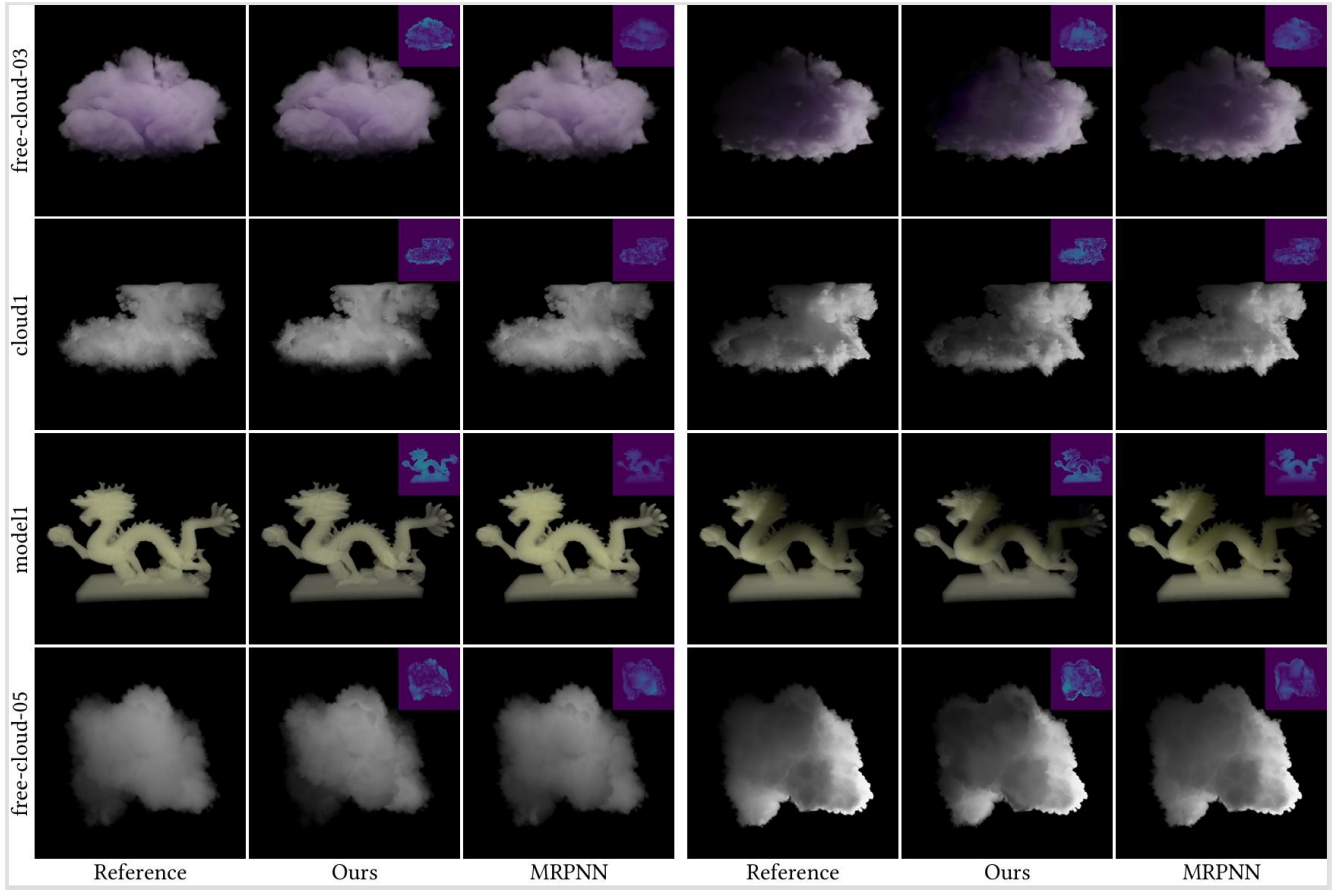
  - Single: same albedo in three color channels;
  - Full: different albedos in three color channels.
  - SH for base grid: do not introduce base grid.
  - **Both of sampling and inference time are significantly accelerated, which finally make our method $4 \sim 5$ times faster than prior SOTA.**

Experiments

- Rendering test:



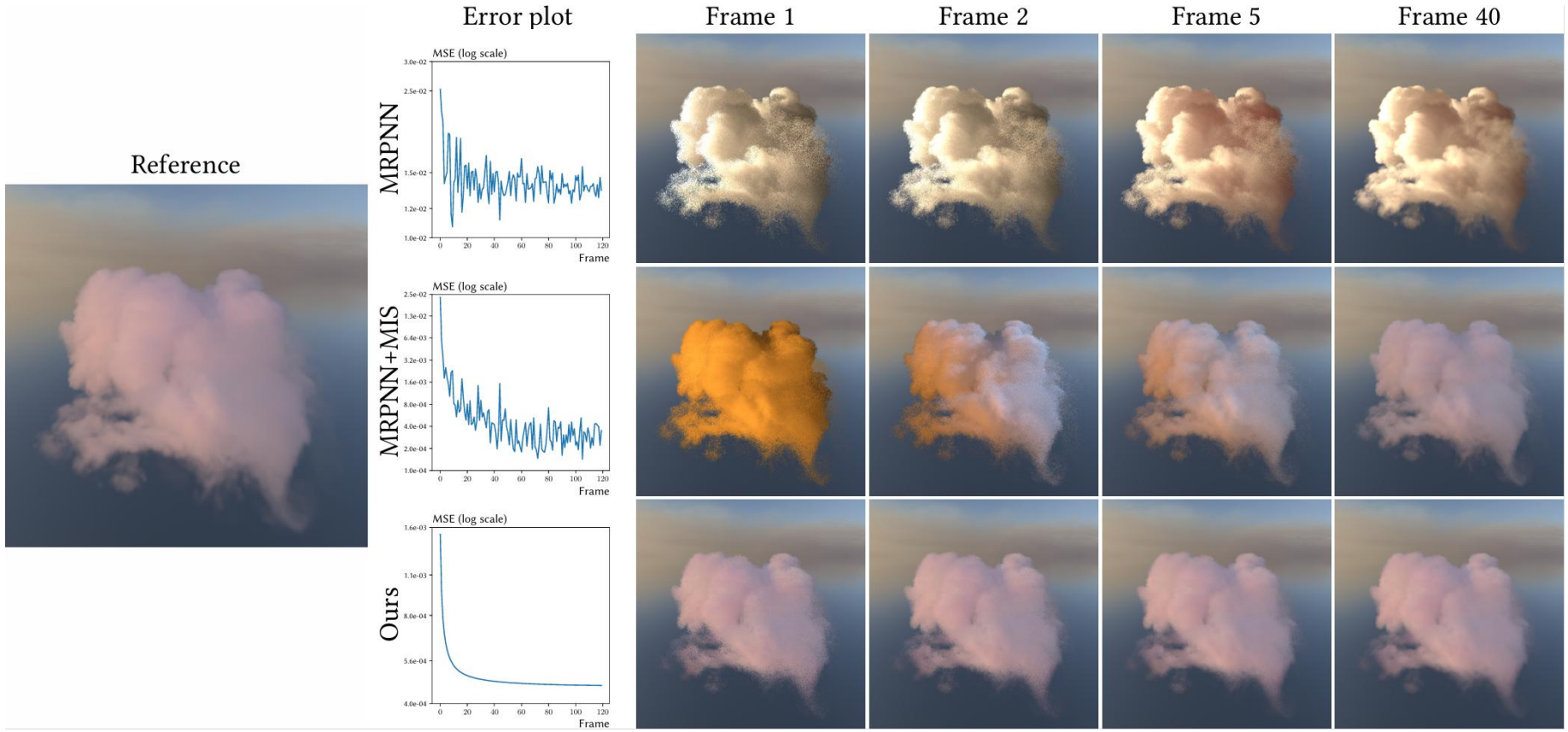
Equal-time rendering (~30 FPS)
Less noisy than SOTA.



Equal-spp rendering
Comparable visual fidelity with SOTA.

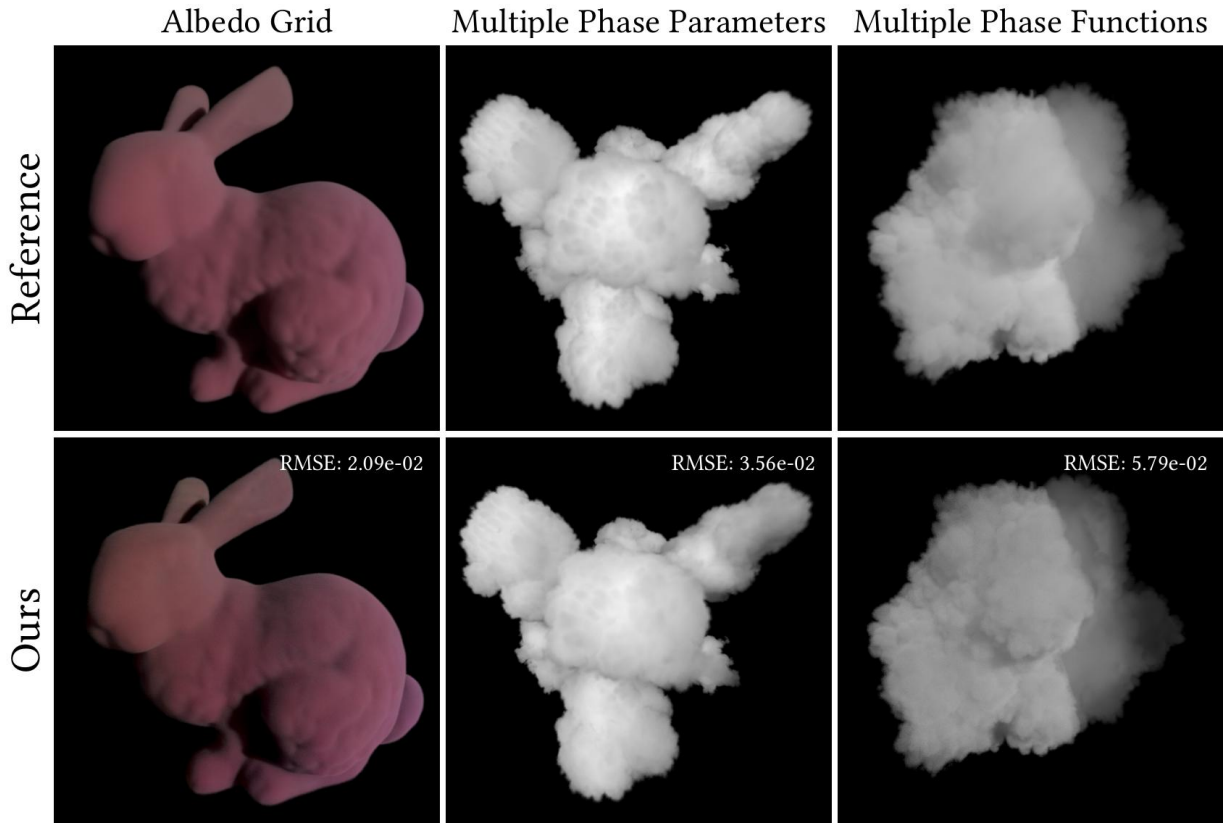
Experiments

- Environment lighting:



Experiments

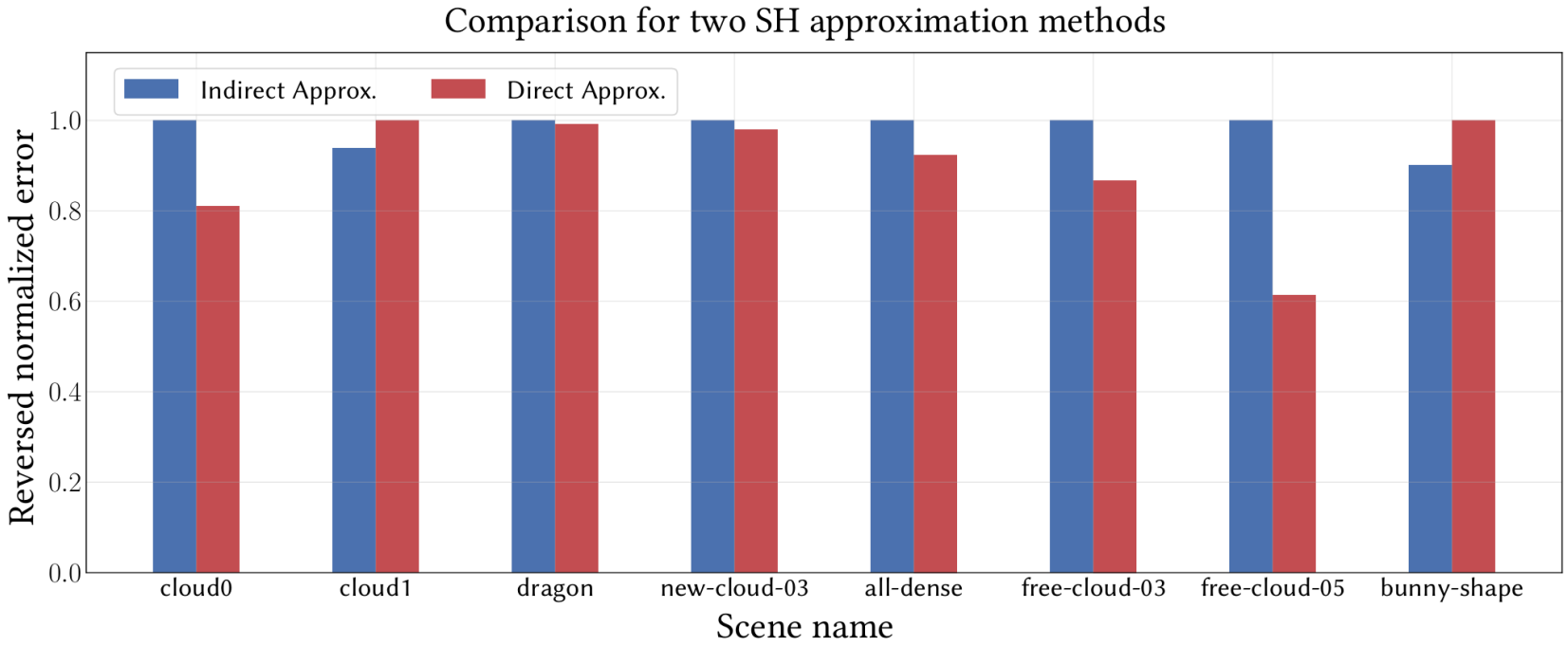
- And other complex settings that cannot be handled previously:



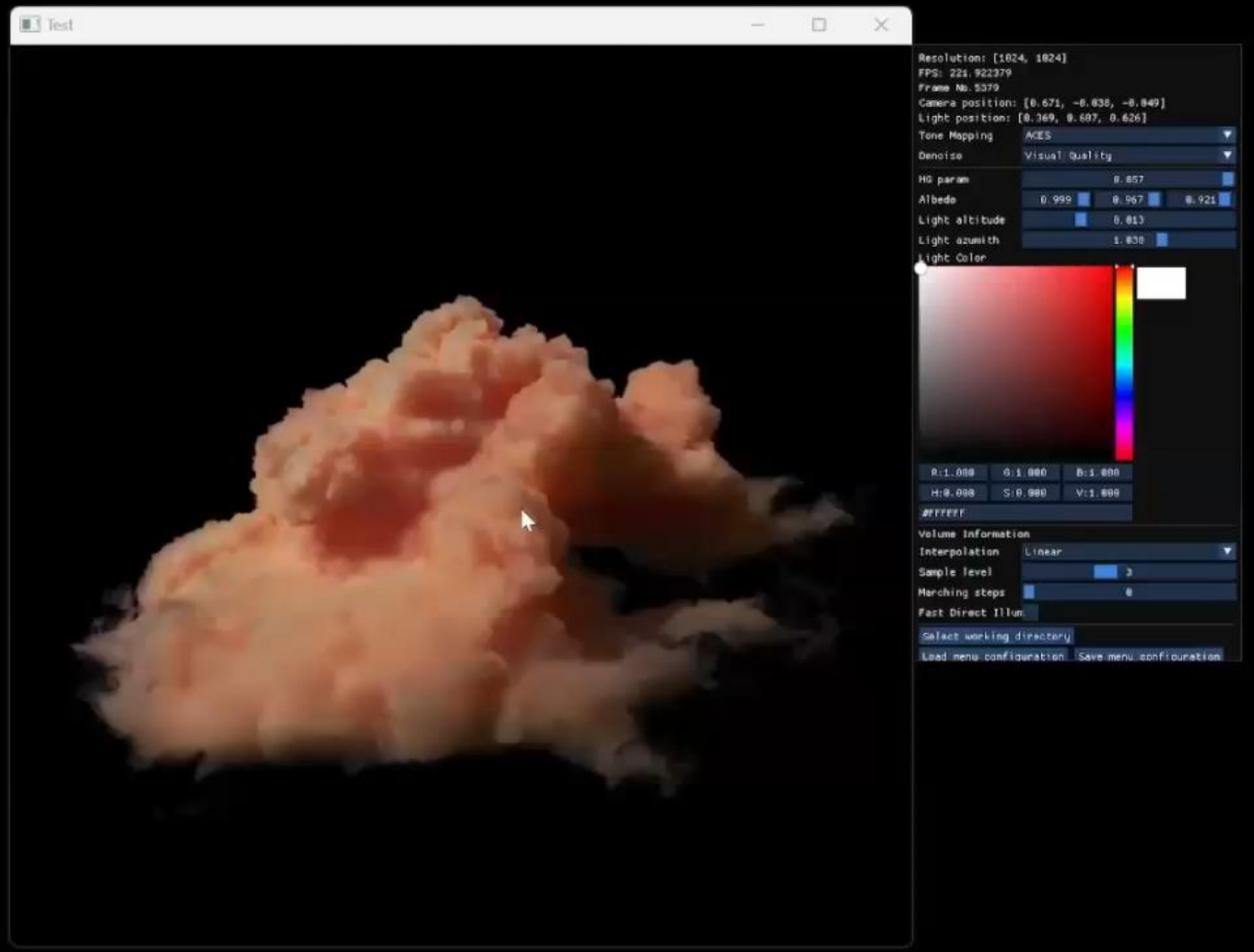
Experiments

- We also conduct an experiment to show effectiveness to apply weighted sum (i.e. $S = \mathcal{S}\left(\sum_{l=0}^{K} \eta_l \sum_m \mathcal{L}_l^m Y_l^m\right)$ vs. $S = \mathcal{S}\left(\sum_{l=0}^{K} \sum_m \mathcal{L}_l^m Y_l^m\right)$):

Higher is better;
In most scenes, applying weighted sum is better.



Resolution: 1024 × 1024,
NVIDIA 4080 SUPER



~210FPS

Post processing:
Only reprojection denoising + tone mapping

Check our supplementary video for details!

Thanks for your listening!

Project page

Github
(Star me!)

Speaker's page
(Contact me for recruitment!)