Deep Illumination-Driven Light Probe Placement

Andreas Tarasidis

Diploma Thesis

Supervisor: Ioannis Fudos

Ioannina, July 2025

Τμήμα Μηχ. Η/Υ & Πληροφορικής

Πανεπιστήμιο Ιωαννίνων

Department of Computer Science & Engineering

University of Ioannina

Dedication

To my mother, father, and brother, who always helped me achieve my goals.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof, Ioannis Fudos, for his invaluable guidance, feedback, and support. As an advisor, he always made time, was extraordinarily patient, and his suggestions were instrumental in the completion of this thesis.

I am also grateful to the friends I've made over the years. Their camaraderie has made my academic journey truly worthwhile.

This journey wouldn't have been possible without them, so finally, but certainly not least, I extend my deepest gratitude to my family. Without their encouragement, boundless support, and unwavering belief in my potential, my works would never have reached fruition.

I dedicate this thesis to them.

Abstract

Realistic lighting is a cornerstone of visually compelling 3D graphics. Unity's Light-Probe system offers an efficient way to capture and interpolate baked Global Illumination (GI) data across dynamic objects in scenes. However, manual placement of light probes in complex scenes is both time-consuming and error-prone, greatly delaying the iteration process when making 3D applications. This thesis presents an automated, deep learning-based approach that predicts per-point importance scores for light probe placement using a PointNet-inspired neural network.

We first generate a regular 3D point grid that conforms to the user-defined arbitrarily-shaped bounds of the scene. We sample per-point lighting information, including spherical harmonics, light-, normal-, and RGB- variance, and occlusion factor as well. These features capture important information that drive GI accuracy. The data is then converted into a concise feature vector at each location, used to then train the PointNet-style AI model that consumes an arbitrary-length list of such feature vectors and outputs a probability in the range 0-1, depicting how vital it is to place a light probe at each point on the grid.

To deploy in Unity, the trained model is then exported to an .ONNX file and imported via Sentis, the official Unity package for handling AI models inside a Unity Runtime; at edit-time, it ingests per-point scene data and returns per-point importance values. Predicted high-importance locations are then used to populate a Unity LightProbeGroup object, giving developers immediate, visually appropriate probe distributions, with easy to control thresh-holding if higher- or lower-importance locations are desired.

We demonstrate that our AI model generalizes across grid sizes and shapes without retraining, as well as giving immediate results for any scene. Although our evaluation remains mostly qualitative, based on visual inspection of GI results and light-probe placement across a variety of indoor and outdoor scenes, we consistently observe that the generated probe layouts capture important scene light-data with minimal or no manual tweaking. By replacing manual probe placement with a simpler AI-based workflow, artists and developers save time and achieve a faster iteration process throughout the development of a 3D application.

Περίληψη

TODO

List of Figures

List of Algorithms

Table of Contents

Chapter 1

Introduction

Modern interactive 3D applications, like video games, VR/AR apps, simulators etc., depend on believable lighting interactions with the objects of a 3D scene to achieve the desired visual goals, while trying to maintain real-time frame-rate budgets, typically above 30 Frames per Second (FPS). Achieving visual fidelity and performance can be a difficult task and sometimes impossible with the given hardware specifications of the device. For that reason, modern real-time rendering engines, e.g. Unity, Unreal Engine, Godot and others, depend on a number of methods to balance those metrics.

The illumination of any scene can be split into two very simple categories. Direct Illumination, the light that travels unoccluded from a light source to a surface of an object, is typically handled with techniques like shadow-mapping or screen-space shadows, yielding crisp, high-framerate-capable shadows, but lack in inter-surface light transport situations. In contrast, Indirect Illumination, or Global Illumination (GI), captures light that has bounced or refracted off one or more surfaces, producing soft shadows, color bleeding, and contextually rich shading.

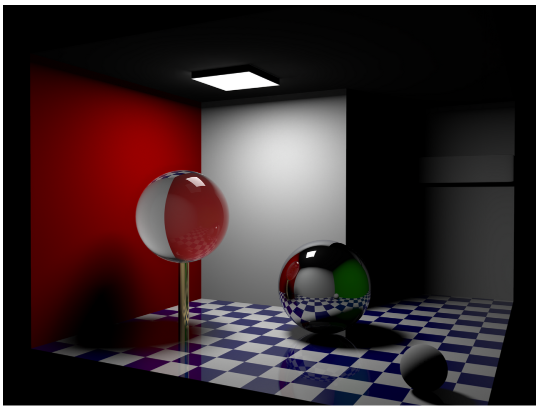

Figure 1.1: Scene lighting with direct illumination only. By Barahag - Own work, CC BY-SA 4.0 https://commons.wikimedia.org/w/index.php?curid=88541991

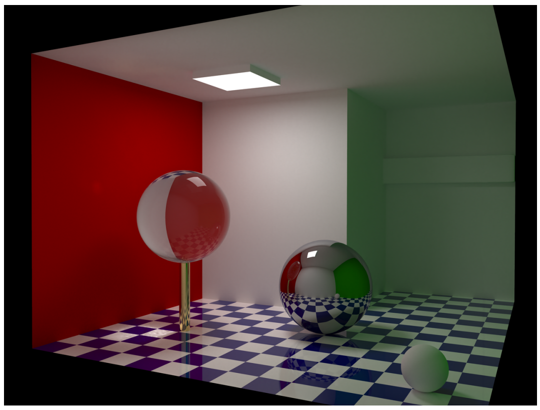

Figure 1.2: Scene lighting with Global illumination. By Barahag - Own work, CC BY-SA 4.0 https://commons.wikimedia.org/w/index.php?curid=88540888

The field-standard for accurate lighting and shadows in a scene is Path-Tracing, a method that tracks every light ray and any interactions it has with the objects of a 3D scene and calculates the resulting color for each pixel of the screen. Such approach remains prohibitively expensive for most interactive applications, so real-time systems employ precomputation and approximation of the illumination of the scene; static geometry is baked into lightmaps that store per-texel irradiance, while dynamic elements sample from irradiance volumes or light probes, sparse 3D points whose spherical-harmonic coefficients are interpolated at runtime.

Screen-space GI methods typically approximate a limited number of light-ray bounces directly from the camera's depth buffer, but suffer from missing contextual information outside the camera's view frustum and temporal instability. Voxel-based approaches (e.g., cone-tracing through a low resolution 3D grid) enable more dynamic multi-bounce effects at the cost of memory, processing cost and potential blurring of fine detail.

Across all these techniques, the central challenge is allocating a strict millisecond-scale budget to indirect illumination while maintaining consistency across static and dynamic scene content, avoiding visible seams when blending baked and runtime solutions and fitting within GPU memory constraints.

Light probes, in particular, represent a compelling middle-ground, flexible enough to illuminate moving objects without rebaking yet compact enough for real-time evaluation, making their optimal placement a critical factor in any high-quality GI pipeline.

1.1 Related Work

There is an abundance of work in the literature addressing the problem of Global Illumination. These studies aim to achieve realistic lighting in 3D scenes by employing various approaches and techniques, each offering unique advantages and disadvantages, but they share a common goal: to maximize visual fidelity while minimizing computational costs.

1.1.1 Offline Methods

Offline Illumination methods refer to techniques that are not viable for real-time applications and are therefore used only in situations where the importance of high visual fidelity far outweighs the need for computational speed, typically in non-interactive 3D renders, most commonly in movies or pre-rendered scenes. Classic Path-Tracing, first introduced in 1986 (Kajiya 1986), tracks the movement of a photon ray emitted from a source, typically the camera, and simulates physics interactions to calculate the color of each screen pixel accurately. The immense computational cost of path-tracing led to the development of performance improvements, such as the Metropolis Light Transport (MLT) method introduced in 1997 (Veach and Guibas 1997), and variants like bi-directional Path-Trace (Lafortune and Willems 1993), which build on Monte-Carlo algorithms (Lafortune 1996).

1.1.2 Online Methods

In contrast, online methods aim to calculate GI interactions in real-time, most commonly used in interactive applications like video games or simulations. They try to balance performance and accuracy, a task that is often difficult due to the processing cost of the calculations for a realistic result. Therefore, these methods take shortcuts, either approximating the GI interactions to a certain degree to maintain framerate budgets, or by precomputing some of the data, wherever possible.

Traditional Methods

Techniques that precompute the illumination of a scene only do so for static geometry; objects in the scene that will never change their position, rotation or scale. The algorithms "bake" the required information onto texture maps, which are rendered as such when needed. Light-mapping is one such technique. It precomputes surface brightness and has a low runtime cost. The game Quake was the first interactive application that used lightmaps for rendering GI (Wikipedia contributors 2025).

Another early technique is the Irradiance Volumes algorithm (Greger et al. 1998), which scatters spherical-harmonic (SH) irradiance samples on a 3D grid on the scene. At runtime, lighting is interpolated from the nearest SH cells; this underlies many probe systems, like Unity's light-probe system that implicitly implements a sparse irradiance volume.

More recent static-GI algorithms include Light Field Probes (McGuire et al. 2017). Light Field Probes extend standard irradiance probes by additionally storing per-texel visibility for each probe. Furthermore, (Xu et al. 2022) introduce Discrete Visibility Fields for static ray-traced lighting. The method precomputes occlusion masks stored in a uniform voxel grid, and at runtime, rays that hit a cell use the stored precomputed masks to quickly cull visibility, skipping geometry already known to be occluded.

Unity's new Adaptive Probe Volumes (APV) build on irradiance volumes by automatically populating a grid, with density matched to local geometry. APV then performs per-pixel probe sampling; each pixel blends from the eight nearest probes (Unity 2025).

Additionally, there are methods that don't focus on Probes for GI. A prevalent example is Unreal Engine's Lumen, a dynamic GI and reflections system that uses a hybrid tracing approach; It starts with a cheap screen-space or signed-distance-field ray cast, and then falls back to more expensive methods like hardware ray tracing (EpicGames 2025).

NVIDIA has also developed RTXGI, a GPU-accelerated library implementing Dynamic Diffuse GI, using a volumetric grid of irradiance probes, which update every frame using hardware-accelerated ray tracing, creating accurate results at the cost of hardware-restricted algorithms and a relatively escalated cost of calculation (Nvidia 2024).

In 2011 (Crassin et al. 2011) , a Voxel Cone Tracing (VCT) technique was introduced to approximate real-time GI. In VCT, the scene's static geometry and lighting are "voxelized" into a 3D texture with multiple levels of mipmapping, containing radiance and opacity. At runtime, indirect illumination is approximated by tracing a few low-resolution "cones" from each surface sample into the aforementioned voxel grid, summing the values from regions of voxels.

Even though there are numerous methods trying to solve real-time GI issues, a big percentage of them tend to revolve around probes of various types; most commonly calculating irradiance values among other high-importance metrics. Therefore, it is vital for a 3D scene to have proper probe placement for best results. There are a few methods that try to automate that process, often by placing the probes in a regular grid and only removing the probes that are inside objects, but that can lead to over-sampling, leading to performance costs, mainly in memory usage budgets. Furthermore, some techniques try to remove additional probes using heuristic methods, therefore approaching optimal placement, but with a significant precomputational cost.

In (Wang et al. 2019), an automatic non-uniform placing scheme is introduced, which uses 3D scene skeletons and gradient-descent refinement to cover important locations without redundant probes. A very recent work formulates geometry-based optimization of probe placement using various mesh features, to further improve the lighting in VR/AR scenarios (Teuber et al. 2024).

Similarly, (Vardis, Vasilakis, and Papaioannou 2021b) approach the problem by starting with a probe set on a dense grid and iteratively removing the least-important probes using radiance error tests, preserving the global light field while minimizing probe count.

AI-based Methods

Recently, AI-assisted methods have started to be developed in order to improve GI in 3D applications, specifically in probe-based solutions. In (Guo et al. 2022), they propose a hybrid neural probe GI. They use a gradient-based search to re-project stored probe radiance for any view, therefore eliminating parallax, and then apply a small neural network to reconstruct high-quality images from low-resolution probe data. Related, a Neural Light Field Probes method has been introduced (You, Geiger, and Chen 2024), which works by decomposing a scene into a grid of trainable light field probes. Each of these probes encodes local radiance and visibility in a compact feature map. Finally, a neural network optimizes these probes so that the summation of their contributions reproduces the full scene lighting.

1.2 Thesis Structure

The structure of the remainder of this thesis will be described shortly. Chapter 2, titled Background, covers important information about light probes and their implementation, describes the AI model basis that was used for our implementation, and introduces the tools and technologies that were used in this thesis. Chapter 3, titled Our Approach, presents our method, describes the implementation of the algorithms used, and explains how each part is combined to create the Light-Probe Neural Network (LPNN), our neural network system that attempts to speed up light probe placement in Unity 3D scenes by predicting importance values for the given grid set and placing only the most vital light probes, affected by a user-controlled threshold value. Each grid position gathers samples of a few metrics, which are then used by the neural network to decide whether or not it is vital to place a light probe in each individual cell of the 3D grid. A step-by-step process of creating the grid, getting the features out of the grid cells, and placing the probes and baking the global illumination inside Unity. Additionally, the feature set can be used to retrain the AI model, we explain how to create the labels needed for the process, and how to import the new model to Unity using Sentis for usage. Chapter 4, titled Experiments, presents an experimental comparative and qualitative evaluation between the proposed method and some of the already introduced algorithms. Finally, Chapter 5, titled Conclusions and Future Works, concludes the thesis and proposes directions for future work based on this thesis.

Chapter 2

Background

In this chapter we introduce some basic, required background for this thesis. First, we introduce light probes and the mathematical equations that define them. Then, we present the AI architecture that was the basis of our AI model. Finally, the tools and technologies used for this thesis are presented.

2.1 Light Probes

As mentioned previously, the idea of using discrete probes to capture scene lighting data traces back to early GI research. In the paper (Greger et al. 1998) introduced the irradiance volume, a 3D grid of sample points storing the irradiance field to approximate GI in complex scenes. A light probe samples the incident radiance at a point in empty space from all directions. Often just the diffuse component of the radiance is captured, since it most commonly varies smoothly, so it can be compactly represented by projecting the lighting onto a truncated spherical harmonic (SH) basis. Third-order SH is most commonly used, storing 9 coefficients per color channel, abbreviated to L2-SH.

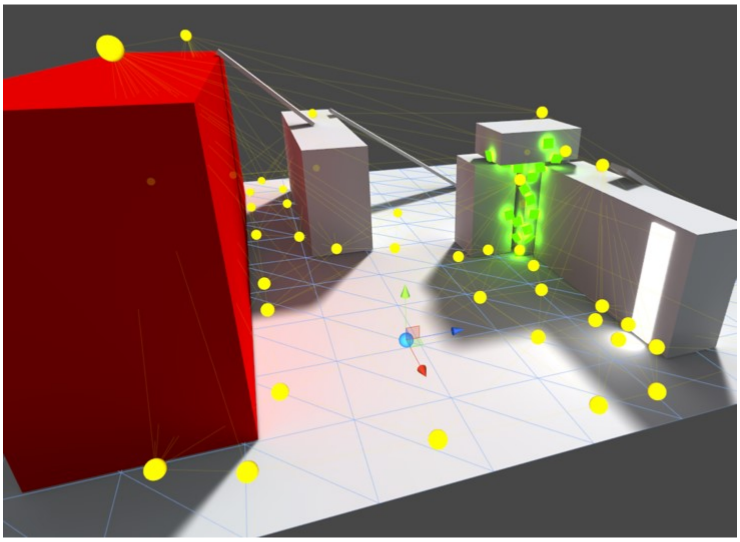

Figure 2.1: A 3D Scene showing a few light probes placed in importance locations (Unity 2016).

2.2 Spherical Harmonics

Spherical Harmonics (SH), first introduced by Pierre Simon de Laplace, are a method of storing information on a point in space. They are categorized in structures called orders. We are interested in third-order SH, since they represent a good middle ground between storage size, computational cost and accuracy. SH are often described as the Fourier Series of functions on the surface of a sphere, breaking down any pattern of light on a sphere into a set of basis frequencies. The order of SH depicts the amount of data we capture, third-order SH, noted as L2 SH, store the first three bands of data, resulting in 9 coefficients per color channel. Bands represent the individual frequencies; the Zeroth band captures the overall average lighting present in that position in space, the First band captures simple directional gradients, and the Second band captures quadratic variations, e.g. gentle light gradients and their shadows.

2.3 PointNet

PointNet is a neural-network architecture designed to work directly on unordered 3D point-clouds; a collection of points without any required grid connectivity. Each point passes through a small MultiLayer Perceptron (MLP), extracting a feature vector that describes its local attributes, e.g. color and normal. After per-point features are computed, PointNet aggregates them into a global descriptor by applying a symmetric operation, usually max-pooling across all points, capturing the strongest signal from the features. Then, the global descriptor is concatenated back to the per-point features, resulting in every point having knowledge about both its characteristics and the broader context. Finally, a per-point MLP refines these combined values into task-specific outputs, commonly classification scores or per-point importance metrics.

Since there is no fixed grid, meaning the points are assumed to be unordered and irregular, traditional CNNs can't be applied directly. Additionally, since PointNet predicts values per-point, it can be used to handle point clouds of any shape and point amount without retraining, limited only by the system's memory. This makes PointNet a fitting candidate to base our model on, with the implementation being the only varying factor.

2.4 Tools

TODO

Chapter 3

Our Approach

what we did and how why etc

Chapter 4

Experiments

In this chapter, we present experimental results of the LPNN approach. The evaluation is qualitative, since the nature of light probes and their optimal placement is subjective to the user and the needs of the application. We focus mainly on speed in relation to light probe layout given by the tool. We compare performance results to LumiProbes (Vardis, Vasilakis, and Papaioannou 2021b). All experiments were conducted in Unity on a system comprising of an NVIDIA RTX2060M GPU, 16GB DDR4 RAM and an Intel i7-9750H CPU, on a Windows 10 Operating System.

4.1 Performance

The LPNN approach speeds up light probe placement by orders of magnitude faster than other approaches, but it may suffer from occasional misplacement; probes that were placed in positions that are not vital, leading to oversampling. In experiments where LumiProbes and LPNN were requested to place close to the same amount of light probes in the same scene, LPNN time stayed close to constant, typically up to a few seconds, regardless of the scenario or the amount requested. Results for various amounts of light probes and scenes are presented in table 4.1. Times are averaged over multiple runs. Units are represented in minutes (m), seconds (s), or milliseconds (ms). Where applicable, we also append the settings used for each tool. For LumiProbes, settings include the grid parameters and the evaluation-point count. All other settings are as follows: Evaluation point placement type is set to Poisson, Decimation type is set to Medium, Decimation directions are averaged, Decimation metric is set to Chrominance, Minimum LP set is disabled, and Maximum Error is set to 3. For LPNN, settings include the threshold value used for the specific result and the cell size of the 3D grid, in order. Figures for the results are shown in section 4.2.

4.2 Quality

The tools were tested on the aforementioned edited Sponza scene (McGuire 2017), Corridor scene (Vardis, Vasilakis, and Papaioannou 2021a) and Office scene (CG AUEB 2021). We will present the qualitative results. As seen in figure **??**, LPNN correctly places light probes in areas of high variance.

Execution time					
Method	Scene	Time	P. Count	P. Present (& Removed)	Settings
LumiProbes	Sponza				
	Office	51.059s	144	84 (60)	(12,3,4), 128
		919.134s	288	182 (106)	(12,3,8), 256
	Corridor	161.151s	180	120 (60)	(20,3,3), 256
		477.648s	243	147 (96)	(27,3,3), 256
Ours	Sponza				
	Office	**7.8ms**	140	34 (106)	0.758, 1.87
		25.2ms	832	117 (715)	0.859, 1.10
	Corridor	**10.9ms**	186	84 (102)	0.549, 1.94
		15.7ms	246	95 (151)	0.615, 1.50

Table 4.1: Execution time for LPNN and LumiProbes on a select number of scenes and probe counts. *P. count* represents the total amount of probes in the scene, before simplification. *P. Present* depict the final amount of probes after running the tools. The number in parenthesis is the amount of probes removed by the tool, in respect to the settings. Fastest times are shown in **bold**.

Chapter 5

Conclusions and Future Work

conclusions and future works

Bibliography

CG AUEB, Group (2021). *Office Unity Scene.* URL: http://graphics.cs.aueb.gr/graphics/index.html.

Crassin, Cyril et al. (2011). "Interactive indirect illumination using voxel-based cone tracing: an insight". In: *ACM SIGGRAPH 2011 Talks.* SIGGRAPH '11. Vancouver, British Columbia, Canada: Association for Computing Machinery. ISBN: 9781450309745. DOI: 10.1145/2037826.2037853. URL: https://doi.org/10.1145/2037826.2037853.

EpicGames (2025). *Lumen GI.* URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine.

Greger, G. et al. (1998). "The irradiance volume". In: *IEEE Computer Graphics and Applications* 18.2, pp. 32–43. DOI: 10.1109/38.656788.

Guo, Jie et al. (Nov. 2022). "Efficient Light Probes for Real-Time Global Illumination". In: *ACM Transactions on Graphics* 41, pp. 1–14. DOI: 10.1145/3550454.3555452.

Kajiya, James T. (Aug. 1986). "The rendering equation". In: *SIGGRAPH Comput. Graph.* 20.4, 143–150. ISSN: 0097-8930. DOI: 10.1145/15886.15902. URL: https://doi.org/10.1145/15886.15902.

Lafortune, E (1996). "Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering". URL: https://web.archive.org/web/20160617231944/http://www.graphics.cornell.edu/~eric/thesis/index.html.

Lafortune, Eric P. and Yves D. Willems (1993). "Bi-directional path tracing". In: *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93).* Alvor, Portugal, pp. 145–153. URL: https://graphics.cs.kuleuven.be/publications/BDPT/.

McGuire, Morgan (2017). *Computer Graphics Archive.* https://casual-effects.com/data. URL: https://casual-effects.com/data.

McGuire, Morgan et al. (2017). "Real-time global illumination using precomputed light field probes". In: *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* I3D '17. San Francisco, California: Association for Computing Machinery. ISBN: 9781450348867. DOI: 10.1145/3023368.3023378. URL: https://doi.org/10.1145/3023368.3023378.

Nvidia (2024). *RTXGI Dynamic Diffuse GI.* URL: https://github.com/NVIDIAGameWorks/RTXGI-DDGI.

Teuber, Maurice et al. (2024). "Geometry-Based Optimization of Light Probe Placement in Virtual Environments". In: *2024 International Conference on Cyberworlds (CW)*, pp. 9–16. DOI: 10.1109/CW64301.2024.00011.

Unity (2016). *Light Probes Example*. URL: https://docs.unity3d.com/530/Documentation/Manual/class-LightProbeGroup.html.

— (2025). *Adaptive Probe Volumes*. URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@17.0/manual/probevolumes-concept.html.

Vardis, Konstantinos, Andreas Vasilakis, and Georgios Papaioannou (2021a). *Corridor Unity Scene*. URL: https://github.com/cgaueb/light_probe_placement/tree/main/Light%20Probes/Assets/Scenes/TestProbesScene.

— (May 2021b). "Illumination-driven Light Probe Placement". In: DOI: 10.2312/egp.20211026.

Veach, Eric and Leonidas J. Guibas (1997). "Metropolis light transport". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., 65–76. ISBN: 0897918967. DOI: 10.1145/258734.258775. URL: https://doi.org/10.1145/258734.258775.

Wang, Yue et al. (2019). "Fast non-uniform radiance probe placement and tracing". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, p. 5.

Wikipedia contributors (2025). *Lightmap — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-June-2025]. URL: https://en.wikipedia.org/w/index.php?title=Lightmap&oldid=1285782125.

Xu, Yang et al. (2022). "Precomputed Discrete Visibility Fields for Real-Time Ray-Traced Environment Lighting." In: *EGSR (ST)*, pp. 81–92. ISBN: 978-3-03868-187-8. DOI: 10.2312/sr.20221158.

You, Zinuo, Andreas Geiger, and Anpei Chen (2024). *NeLF-Pro: Neural Light Field Probes for Multi-Scale Novel View Synthesis*. arXiv: 2312.13328 [cs.CV]. URL: https://arxiv.org/abs/2312.13328.