ABLE-NeRF: Attention-Based Rendering with Learnable Embeddings for Neural Radiance Field

Zhe Jun Tang[1] Tat-Jen Cham[2] Haiyu Zhao[3]

[1]S-Lab, Nanyang Technological University [2]Nanyang Technological University
[3]SenseTime Research

Abstract

Neural Radiance Field (NeRF) is a popular method in representing 3D scenes by optimising a continuous volumetric scene function. Its large success which lies in applying volumetric rendering (VR) is also its Achilles' heel in producing view-dependent effects. As a consequence, glossy and transparent surfaces often appear murky. A remedy to reduce these artefacts is to constrain this VR equation by excluding volumes with back-facing normal. While this approach has some success in rendering glossy surfaces, translucent objects are still poorly represented. In this paper, we present an alternative to the physics-based VR approach by introducing a self-attention-based framework on volumes along a ray. In addition, inspired by modern game engines which utilise Light Probes to store local lighting passing through the scene, we incorporate Learnable Embeddings to capture view dependent effects within the scene. Our method, which we call ABLE-NeRF, significantly reduces 'blurry' glossy surfaces in rendering and produces realistic translucent surfaces which lack in prior art. In the Blender dataset, ABLE-NeRF achieves SOTA results and surpasses Ref-NeRF in all 3 image quality metrics PSNR, SSIM, LPIPS.

1. Introduction

Neural Radiance Field (NeRF) has become the de facto method for 3D scene representation. By representing the scene as a continuous function, NeRF is able to generate photo-realistic novel view images by marching camera rays through the scene. NeRF first samples a set of 3D points along a camera ray and outputs its outgoing radiance. The final pixel colour of a camera ray is then computed using volumetric rendering (VR) which colours are alpha-composited. This simple approach allows NeRF to generate impressive photo-realistic novel views of a complex 3D scene. However, NeRF is unable to produce accurate colours of objects with view-dependent effects. Colours of translucent objects often appear murky and glossy objects have blurry specular highlights. Our work aims to reduce these artefacts.

Figure 1. We illustrate two views of the Blender 'Drums' Scene. The surface of the drums exhibit either a translucent surface or a reflective surface at different angles. As shown, Ref-NeRF model has severe difficulties interpolating between the translucent and reflective surfaces as the viewing angle changes. Our method demonstrates its superiority over NeRF rendering models by producing such accurate view-dependent effects. In addition, the specularity of the cymbals are rendered much closer to ground truth compared to Ref-NeRF.

The exhibited artefacts of the NeRF rendering model is largely due to the inherent usage of VR as features are ac-

cumulated in the colour space. Variants of NeRF attempt to tackle this defect by altering the basis of this VR equation. For instance, Ref-NeRF first predicts the normal vector of each point on the ray. If a point has a predicted normal facing backwards from the camera, its colour is excluded from computation via regularisation. However, prediction of normals in an object's interior is ill-posed since these points are not on actual surfaces. As a consequence, Ref-NeRF achieves some success over the baseline NeRF model, albeit imperfectly.

When rendering translucent objects with *additional specular effects*, NeRF and its variants suffer from the same deficiency. This is due to the computation of σ which is analogous to the 'opacity' attribute of a point used in the VR equation. It is also related to the point's transmissivity and its contribution of radiance of to its ray. As per the Fresnel effect [5], this property should depend on viewing angles. Similarly, [19] describes a notion of *'alphasphere'*, which describes an opacity hull of a point that stores an opacity value viewed at direction ω. Most NeRF methods disregard the viewing angle in computing σ. In fig. 1, the surface of the uttermost right drum in the Blender scene exhibits changing reflective and translucent properties at different viewing angles. Ref-Nerf and other variants, by discounting the dependency of σ on viewing angle, may not render accurate colours of such objects.

Additionally, learning to model opacity and colour separately may be inadequate in predicting the ray's colour. Accumulating high-frequency features directly in the colour space causes the model to be sensitive to both opacity and sampling intervals of points along the ray. Therefore we rework how volumetric rendering can be applied to view synthesis. Inspecting the VR equation reveals that this methodology is similar to a self-attention mechanism; a point's contribution to its ray colour is dependent on points lying in-front of it. By this principle we designed ABLE-NeRF as an attention-based framework. To mimic the VR equation, mask attention is applied to points, preventing them from attending to others behind it.

The second stage of ABLE-NeRF takes inspiration from modern game engines in relighting objects by invoking a form of memorisation framework called 'baking'. In practice, traditional computer graphics rendering methods would capture indirect lighting by applying Monte Carlo path tracing to cache irradiance and then apply interpolation during run-time. Similarly, game engines would use lightmaps to cache global illumination for lower computational costs. For relighting dynamic objects, localised light probes are embedded in the scene to capture light passing through free space. At run-time, moving objects query from these light probes for accurate relighting. The commonality between all these approaches is the process of 'memorising' lighting information and interpolating them during run time for accurate relighting. As such, we take inspiration from these methods by creating a memorisation network for view synthesis. Given a static scene, we incorporate Learnable Embeddings (LE), which are learnable memory tokens, to store scene information in latent space during training. Specifically, the LE attends to points sampled during ray casting via cross-attention to memorise scene information. To render accurate view dependent effects a directional view token, comprising of camera pose, would decode from these embeddings.

ABLE-NeRF provides high quality rendering on novel view synthesis tasks. The memorisation network achieves significant improvements in producing precise specular effects over Ref-NeRF. Moreover, by reworking volumetric rendering as an attention framework, ABLE-NeRF renders much more accurate colours of translucent objects than prior art. On the blender dataset, ABLE-NeRF excels both quantitatively and qualitatively relative to Ref-NeRF.

In summary, our technical contributions are:

(1) An approach demonstrating the capability and superiority of transformers modelling a physics based volumetric rendering approach.

(2) A memorisation based framework with Learnable Embeddings (LE) to capture and render detailed view-dependent effects with a cross-attention network.

2. Related Work

We first review techniques from computer graphics for capturing indirect lighting effects and global illumination. Following, we discuss how NeRF and its other variants render photo-realistic images of a 3D scene from an unseen viewing angle.

Indirect Illumination in Rendering. Rendering with indirect illumination is a widely studied topic. Pioneering works using path tracing [15] or light tracing [10] cast rays from a camera until they hit a point and traces random rays at the visible surface to light sources. However, these methods requires heavy computation as sampling multiple rays is a costly operation. Instead, irradiance caching [16] is applied to sparsely samples rays and its indirect illumination is stored to speed up this process. An object's illumination will then be interpolated at its nearby cached values. Other methods involving a pre-computation based method like radiance transfer and lightmaps [1], first calculate the surface brightness and store it in texture maps for real time performance. Unlike lightmaps storing surface lighting information, light probes [27] bake lighting information passing through the scene. During run time, dynamic objects would query from the nearest light probes for indirect lighting information. The use of probes can be similarly be extended to reflections. In game engines, reflection probes [28] are made to capture images as cubemaps within the scene. These cubemaps are then utilised by objects with

reflective materials to produce convincing reflections of the environment.

The impetus to incorporate Learnable Embeddings in our work takes inspiration from how these light or reflection probes function. Yet, our work differs from the traditional graphics pipeline in the type of information being captured. Unlike probes in game engines, these embeddings do not exist as physical entities in the 3D scene geometry. Instead, Learnable Embeddings operate within the latent space as learnable latent vectors. In this manner, the LE capture latent information of a given scene. Thus, it is crucial to optimise these LE via training. Similar to relighting dynamic objects by interpolating nearby light probes or reflection probes, new viewing angles would query from these LE to achieve accurate view dependent effects.

3D Scene Representation for View Synthesis Numerous methods have been proposed for generating new images of a scene using only a few captured images. Light field rendering methods [13, 18] characterise the unobstructed flow of light passing through the scene as 4D function and slice this slab differently to generate new views. While these methods require a large number of light field samples to interpolate new views, recent deep learning-based methods [23] only require sparse inputs. Separately, image based rendering methods [6, 7, 14, 26, 31] balance a set of weights heuristically or learned to blend nearby input images creating novel views. Scene representation methods also extend to volumetric methods by colouring voxel grids [22] or reconstructing plenoxels [12]. Methods involving neural networks are also capable of learning volumetric scene representation through gradient-based methods [11, 17, 24, 25]

The shift towards coordinate-based methods has shown a quantum leap in concise 3D scene representation. With a few layers of MLP, NeRF [20] can map a continuous input of 3D coordinates to the scene geometry and appearance. NeRF can also be extended to dynamic scenes, avatar animations, and even scene editing. These algorithms, which model appearance, typically decompose scene materials into its BRDF properties [34]. As a result, they require strong physics based assumptions such as known lighting conditions or single type materials. On the contrary, Ref-NeRF [30] does not assume these precise physical meanings. This enables Ref-NeRF to avoid relying on such assumptions. Our work follows this school of thought. We do not assume a physics based learning approach as we replace volumetric rendering by an end to end deep learning methodology.

Transformers for View Synthesis The use of transformers for view synthesis have gained popularity lately. IBRNet [31] applies a ray transformer to generate σ values before using the VR equation to accumulate colours. In [26], the authors apply a two-stage transformer-based model to aggregate features along epipolar lines in each reference views and then combine these reference views with a view transformer. SRT [21] extracts features from training images with a CNN and then apply transformers to aggregate features before using a target ray to query for a pixel colour. NeRF-in-detail [2] also uses a transformer to propose new sample points along a ray and then apply NeRF to generate a ray colour. Unlike ABLE-NeRF, none of these methods apply transformers to model a physics based volumetric rendering approach.

2.1. Neural Radiance Field Overview

NeRF represents a 3D scene as a continuous volumetric scene function. It traces a pixel ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, into a scene where $\mathbf{o}$ and $\mathbf{d}$ represent the camera origin and pose. After sampling for 3D points along the ray, NeRF predict point's opacity using spatial MLPs. Following which, a directional MLP determines the colour of the point. Finally, to compute the colour of a ray, alpha composition with numerical quadrature is applied to these points based on (1).

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\sigma_i \delta_i\right)\right) \mathbf{c}_i \tag{1}$$

where

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \tag{2}$$

NeRF maintains two separate sets of model parameters for the coarse and fine network. The network is optimised with a total squared error between the predicted pixel colour and the true pixel colours of both the coarse and fine network.

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C_c(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C_f(\mathbf{r}) \right\|_2^2 \right] \tag{3}$$

In practise, only the output of the fine network is used to render the final image.

3. Method

As aforementioned, applying NeRF's volumetric rendering to accumulate features in the colour space causes the outgoing radiance to be highly sensitive to both opacity σ prediction and the point sampling intervals δ. Despite the δ intervals, the density σ of each point acts as a differential opacity for controlling the accumulated radiance along a ray passing through space [20]. As such, NeRF has difficulty predicting the colour of a surface point exhibiting both transmissive and reflective properties at different angles, resulting in a 'murky' appearance. ABLE-NeRF addresses this issue by diverging from such physics-based volumetric rendering equation. Instead, we formu-

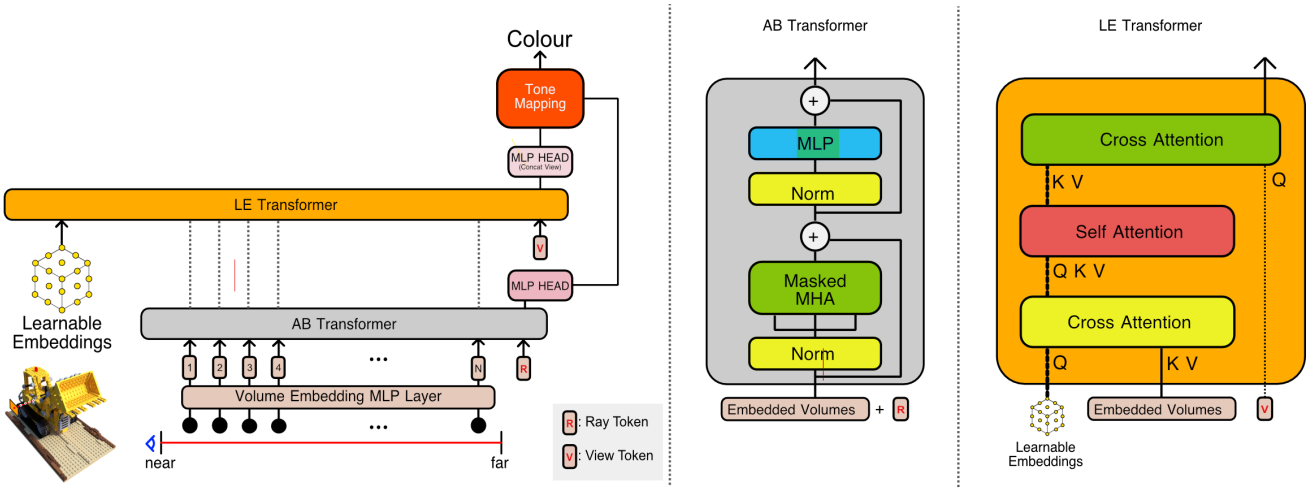

Figure 2. A visualisation of ABLE-NeRF. Similar to mip-NeRF, we cast a ray and sample for N conic frustum volumes between the near and far boundary. Each volume passes through a Volume Embedding layer consisting of several layers of MLP. A ray token 'R' is appended to the sequence of points before propagating it to the Attention-Based rendering Transformer (AB Transformer) module. After the last transformer layer, the ray token is used to compute a non explicit view-dependent colour. Next, several Learnable Embedding (LE) and a view-dependent token 'V' are appended to the sequence of embedded volumes post AB Transformer module before passing to LE Transformer. Within LE Transformer, LE cross-attend to the embedded volumes to memorise static scene information. LE then processes this information with self-attention and a view-dependent token 'V' decodes from LE. The final colour is produced by a tone mapping function that takes into account both the colour and view tokens, after the MLP head.

late an attention-based network in ABLE-NeRF to determine a ray's colour. These changes allow ABLE-NeRF a flexibility to selectively assign attention weights to points compared to alpha compositing point features (2) along a ray. We constrained the attention mechanism by introducing masks where frontal points are restricted from attending to rear points. This masking strategy allows us to encode a viewing directional information implicitly. In addition, to capture view-dependent appearance caused by forms of indirect illumination, we incorporate LE as a methodology inspired by light and reflection probes from modern game engines.

3.1. Attention-based Volumetric Rendering

NeRF predicts both σ value and colour of a sampled point. As a consequence, NeRF faces difficulties in predicting a surface that exhibits both translucent and reflective properties at different angles shown in fig. 1. Authors of [30] attribute NeRF's inadequacy in predicting an object's specular effects to the difficulty in interpolating between 'highly complicated functions' for glossy appearance. We further extend this argument, stating it is even more challenging to interpolate between glossy and translucent appearances of a sampled point that exudes *both* characteristics.

To solve this issue, we can decompose the problem into rendering translucent and reflective surfaces separately. Determining a point's σ is equivalent to controlling a point's opacity [20]. Therefore, points along a translucent surface should have low σ values to describe a low radiance accumulation along a ray. Conversely, for an entirely reflective surface, the points of the reflective surface should have a high σ value to indicate a high outgoing radiance. Thus, predicting a point's σ is critical in describing its outgoing radiance. However, in NeRF, σ is fixed for a point that is either translucent or reflective at different angles. In this scenario, the task of predicting a point's outgoing radiance is left to the viewing directional MLP, which is ill-equipped to do so.

Inspired by the use of volumetric rendering (2), the weight of a point depends on the weights of itself and the frontal points lying along the same ray. In our work, we apply a transformer model to generate the weights of individual points of the same ray. With this approach, we do not generate σ values directly based on the spatial position of a sampled conic frustum of volume [3]. Instead of assigning weights based on σ and δ as per (1), the importance of a point contributing to a ray's radiance is determined by an attention mechanism.

For a given ray, we sample N number of conic frustums of volumes along it encoded with Integrated Positional Encoding (IPE) described in mip-NeRF. Each conic volume passes through a volume embedding block of four MLP layers to generate a volume embedding $\mathbf{v}^i$, where i denotes the position of conic volume along the ray starting from the camera, with latent dimensional size of D. Similar to

ViT [9] and BERT's [class] token [8], we prepend a ray token $\mathbf{R}$ of the same dimension to the sequence of volume embeddings. We abuse the notation of sets to describe an input sequence $\mathbb{Z}_0$, as a set of ray token and the sequence of embedded conic volumes, to the first transformer layer as described in (4). The subscript notation in $\mathbb{Z}$ is used to denote the number of successive self-attention operations on the set.

In a manner similar to (2), we utilise a 'masking' technique to the limit the attention of volume embeddings solely to those that lie ahead of them along the ray, thereby excluding all others. Specifically, a volume embedding can only attend to the itself, the ray token, and other volume embeddings lying in front of it. This exclusion is represented in the set exclusion shown in (5), where conic volumes sequenced behind $\mathbf{v}^i$ are excluded from the standard self attention operation. The masking is expressed by setting the scaled-dot product attention to $-\infty$ in the input of softmax, similar to the decoder settings of Transformers, to denote illegal connections [29]. This 'masking' constraint allows us to implicitly encode view-dependent information; zero masking indicating a bi-directional ray, while masking constraints it to being uni-directional. We demonstrate in sec. 5.1 the importance of masking. No masking is applied to the ray token (6).

After the final encoder layer L, a single MLP classification head is attached to $\mathbf{R}_L$ to predict the colour of the ray (7). The equations are presented below.

$$\mathbb{Z}_0 = \{\mathbf{R}, \mathbf{v}^1, \mathbf{v}^2, ..., \mathbf{v}^N\} \tag{4}$$

$$\mathbf{v}_l^i = \text{Att}(\mathbb{Z}_{l-1} \backslash \{\mathbf{v}_{l-1}^{i+1}, ..., \mathbf{v}_{l-1}^N\}) \tag{5}$$

$$\mathbf{R}_l = \text{Att}(\mathbb{Z}_{l-1}) \tag{6}$$

$$\mathbf{y} = \text{MLP}(\mathbf{R}_L) \tag{7}$$

3.2. Hierarchical Volume Sampling with Coarse-Fine Feature Propagation

We follow the general NeRF rendering strategy in creating two networks: coarse and fine. In NeRF, the coarse network uses N_c stratified samples as inputs and then resamples $N_f = \frac{1}{2}N_c$ points. Next, the fine network uses the total $N_c + N_f$ points to produce the final colour. Unlike NeRF, mip-NeRF samples $N_c = N_f$ conic frustum volumes for each of the coarse and fine networks. The final predicted ray colour uses only N_f samples for computation, discarding information from the coarse network. In our work, the coarse network also uses N_c stratified samples. To generate $N_f = N_c$ samples in our fine network, we sample from the attention weights of the output coarse ray token at state $\mathbf{R}_L^C$ (after L layers attending to all the coarse volume embeddings in the coarse network). Unlike mip-NeRF which discards coarse sample information entirely, we retain this information by reusing coarse ray token as the input fine ray token ($\mathbf{R}_0^F = \mathbf{R}_L^C$) for the fine network. Thus, we retain the ray representation from the coarse network. This approach allows us to avoid the quadratic cost of scaling up to an entire $N_c + N_f$ samples in every transformer layer of the fine network and only rely on N_f samples.

3.3. Learnable Embeddings for View-Dependent Appearance

NeRF's rendering process strictly calculates the radiance of points along a ray. However, the directional MLP is insufficient in computing the specularities of a point. Other NeRF variants attempt to resolve this with some success by predicting a reflection direction of each point [30]. The general rendering equation [15] describes how indirect illumination should include multi bounce lighting, where lights reflects off surfaces and shines upon other surfaces. In NeRF's strict rendering ray casting approach, only points on the ray are used for radiance computation. Consequently, NeRF's rendering model can only coarsely approximate direct and indirect illumination using a view direction. We are interested in resolving this issue by capturing the indirect illumination effects radiated by other possible sources. Hence, it is imperative to formulate a query process for external sources *beyond volumes along a ray*. Inspired by game engines' usage of probes, we create LE to store static scene information. These LE serves as a form of memory which allows us to design a secondary branch of attention mechanism as seen in fig. 2.

Like the ViT class token, Learnable Embeddings (LE) in our work are trainable network parameters (memory tokens) used to capture static lighting information by querying from conic frustums in latent space. The iterative training process whereby LE attends to conic volumes in the scene allows the scene lighting information to be encoded as memory. During inference, conic volumes are mapped into latent space via these embeddings and then decoded with a view directional token. In our architecture, the view token is a camera pose Fourier encoded by 16 bands and mapped to the same dimension as LE via a linear layer.

3.4. Tone Mapping

The attention-based rendering backbone outputs the direct illumination exuded by the conic frustum of volumes along the ray. Separately, the cross-attention branch with LE outputs the view dependent illumination of these volumes. In this manner, we prevent the network from overfitting with this separation. To combine both the outputs, we apply a fixed mapping function to convert linear colours to sRGB, capped to [0,1] as Ref-NeRF [30].

4. Experiments

We implement our model on two datasets; the Blender dataset and Shiny Blender dataset. Similar to mip-Nerf [3],

Model	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG	20.60	0.861	0.144
VolSDF	27.96	0.932	0.096
NSVF	31.74	0.953	0.046
NeRF	32.38	0.957	0.043
Mip-NeRF	33.09	0.961	0.043
Ref-NeRF	33.99	0.966	0.038
Ours, no LE	34.05	0.963	0.041
Ours	35.02	0.975	0.035

Table 1. Baseline comparisons of ABLE-NeRF and previous approaches on Blender dataset. Results extracted from [30].

we sample conic frustums of volumes along a ray. Our model maintains two networks, coarse and fine. The number of transformer layers, L, described in sec. 3.1 is 2 and 6 for the coarse and fine networks respectively. The coarse network is designed as a lighter network with fewer layers, as its purpose is to generate fine samples, similar to mip-NeRF 360 [4] proposal MLP. We sample 192 conic frustums in total, 96 samples in each network, and included 32 LE (shared by coarse and fine networks) to store view-dependent information. The volume embedding module consists of 4 MLP layers, each with 192 hidden units, with ReLU activations. The dimensions of each transformers are set to 192, the same dimension as the volume embedding layers and in the feed-forward layers, the FF hidden unit ratios are set to 1:3. For the Shiny Blender dataset, we set the number of LE to 16, as it contains simpler objects compared to the standard Blender dataset. Optimisation on each scene is trained for 250k iterations.

On each dataset, we evaluate ABLE-NeRF with three commonly used image quality metrics; PSNR, SSIM, LPIPS. A full breakdown of per-scene scores is included in the supplementary materials.

4.1. Blender Dataset

We compare ABLE-NeRF with the latest neural based synthesis network on the standard Blender dataset that originated from NeRF's paper. The results in Table 1 shows that our work surpasses prior art when compared to the top performing NeRF based method which applies a physics-based volumetric rendering.

ABLE-NeRF also outperforms prior art qualitatively in rendering photo-realistic appearances of surfaces. As seen in fig. 3, ABLE-NeRF renders compelling visuals of highly complex surfaces in the Blender Ship scene where the surfaces of the waves resemble the ground truth more closely compared to Ref-NeRF. In the Materials scene, ABLE-NeRF produces reflections of intra-scene objects, attributed to the use of LE, which captures multi-bounce lighting effects. The appearance of reflections of spheres off another neighbouring sphere (reflections of reflections) is clearer

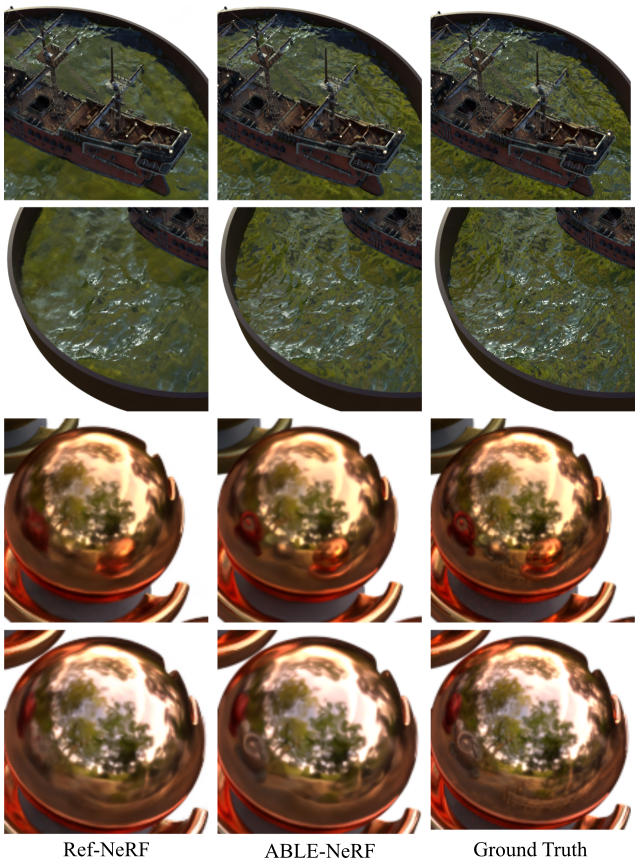

Figure 3. ABLE-NeRF significantly improves upon visual realism of highly complex surfaces such as the waves in the Blender ship scene. Furthermore ABLE-NeRF is able to capture intra-scene reflections of neighbouring spheres off glossy sphere in the Blender Materials scene. Top performing NeRF based variant often fail in producing surfaces of complex geometries and challenging view-dependent multi-bounce lighting.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG	26.21	0.921	0.121
Mip-NeRF	29.21	0.942	0.092
Ref-NeRF (no pred. normals)	30.91	0.936	0.105
Ref-NeRF	35.96	0.967	0.058
Ours	33.88	0.969	0.075

Table 2. Baseline comparisons of ABLE-NeRF and previous approaches on Shiny Blender dataset. Results extracted from [30].

compared to standard ray-casting approaches of NeRF. This highlights the importance of maintaining LE to capture indirect lighting effects.

4.2. Shiny Blender Dataset

Compared to the Blender dataset by NeRF [20], the Shiny Blender dataset by Ref-NeRF [30] contains objects with more glossy surfaces. It is important to note that the

Figure 4. ABLE-NeRF is able render intra-scene surface reflections better than Ref-NeRF. As shown in the Shiny Blender Coffee scene, the reflection of the teaspoon on the side of the cup appears more apparent than Ref-NeRF.

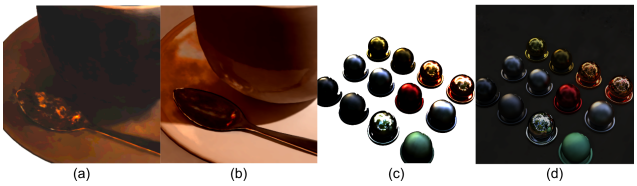

Figure 5. Prior to tone-mapping, (a) and (c) are outputs from the AB Transformer while (b) and (d) are outputs from the LE Transformer.

Shiny Blender dataset mostly consists of singular objects with simple geometries, where the surfaces are smooth and rounded. As a result, Ref-NeRF outperforms ABLE-NeRF in terms of PSNR and LPIPS since the normals for such objects are easier to predict, compared to the complex surfaces in the standard Blender dataset of NeRF. For example, for a smooth rounded 'Shiny Ball', Ref-NeRF outperforms ABLE-NeRF due to its simpler geometry. However, for a more complex surface such as the 'Toaster', ABLE-NeRF outperforms Ref-NeRF. We display Ref-NeRF ablation study results with no normal predictions to support our case. Without normal predictions, ABLE-NeRF surpasses Ref-NeRF by a wide margin.

It is worth highlighting to readers that ABLE-NeRF excels at capturing intra-scene reflections of surfaces caused by multi-bounce lighting, which are highly complex scenes. In such scenarios, the reflections of objects interact with other objects within the scene. In fig. 4, we show reflections of the teaspoon off the cup in the 'Coffee' scene is rendered closely to ground truth. Ref-NeRF fails to capture such intra-scene reflections compared to ABLE-NeRF. The intra-scene reflection due to multi-bounce lighting is well captured as shown in fig. 5.

5. Architectural Analysis

5.1. Masking Strategy

The masking strategy is imperative in allowing transformers to model volumetric rendering as an end to end deep learning based approach. Without masks, the model would render inaccurate surfaces as seen in fig. 6. By including masks, we implicitly encode a uni-directional ray versus a non-masking bidirectional ray as shown in the figure. We have attempted to introduce a uni-directional ray information without masks by appending a volume's position on a ray. However, this attempt is ineffective compared to our original masking strategy.

Figure 6. Here we show the importance of masking rear points from frontal point along a camera ray. Without mask, a bi-directional ray is implied causing the network to have difficulty rendering the object's surface accurately. With the masking strategy, we enable transformers to mimic a volumetric rendering strategy and also implicitly encode a view directional information.

Figure 7. As we corrupt the weights of LE with additive Gaussian noise, we observe that the view-dependent surfaces of the drum scene changes. As we continue to destroy the weights of LE by setting it to zero, we corrupt the specularities and transparencies of the drums. Observe that the specularity in left cymbal of the uttermost right figure is completely eradicated. Diffuse surfaces largely remain unchanged perceptually.

5.2. Learnable Embeddings

Learnable Embeddings Inclusion We validate our model design choices by performing an ablation study on the Blender dataset. In this setup, we exclude LE and compute the final ray colour using only the ray token. The results are included in Table 1. Without LE, the model performs comparatively with Ref-NeRF. By including LE, ABLE-NeRF has the flexibility to attend to embeddings not constrained to the line of ray. Evidently, we demonstrate in fig. 8 that LE allows our model to capture better view-dependent specularities.

Perturbating Learnable Embedding We formally described LE as a form of memory for a static scene. As a form of analysis to understand what LE are effectively memorising, we perturbed these memory by adding Gaus-

Ground Truth | Complete Model | No Learnable Embeddings | Ref-NeRF

Figure 8. We visualise how ABLE-NeRF benefits from the use of transformers for volumetric rendering and also the inclusion of LE. Observe the specularities of the right and centre cymbals. Removing LE causes the model to fail in capturing specular effects effectively on the cymbals. Even without LE, the basic backbone of using transformers as a deep learning VR-based approach allows us to render the translucent portions of the drums more accurately compared to NeRF based approaches.

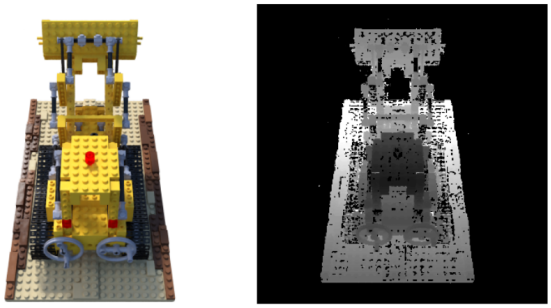

Figure 9. We extract the per-volume attention from the ray token. By selecting the highest attention weight to the volume, we are able to plot a depth map based on the distance traversed to that conic volume frustum along the ray from the camera origin.

sian noise to corrupt these memory tokens. Lastly, we wiped the memory of LE, collapsing all memory into a single entity, by setting its weights to zero. With reference to fig. 7, observe that the diffuse surfaces remains perceptually unchanged while view-dependent surfaces with specularities and translucency are affected. This analysis offers us insights to how LE could be modified to edit scenes dependent on viewing direction for future work.

5.3. Attention Maps as Depth Maps

As the ray token selectively assigns attention weights to conic volume frustums along the ray, the volume with the highest attention weight could imply a surface of the object in the scene. With the attention map, we plot a depth map based on the distance traversed from the camera origin to the volume with the highest attention weight. Fig. 9 illustrates the capability of ABLE-NeRF in generating depth maps from attention weights.

6. Conclusion

We have highlighted the general issues of NeRF-based rendering methods that use physics-based volumetric rendering, which cause difficulties in rendering complex surfaces that exhibits both translucent and specular appearances. Our model, ABLE-NeRF, ameliorates such issues by applying a deep learning-based method using masking on transformers to learn a physics-based volumetric rendering method. With the attention weights generated by transformers, we can re-sample a 3D space effectively with visual content and output a depth map from an attention map. Lastly, we have included Learnable Embeddings as a form of memorisation framework to capture view-dependent lighting effects in latent space and allow the view angle token to query these LE beyond a ray for accurate view-dependent visuals. These contributions allow ABLE-NeRF to significantly improve upon prior art in novel view synthesis. We believe that our work paves the way forward in rendering accurate visuals of complex objects and scenes, as well as hinting at the potential for new scene editing methods by reprogramming LE. Our code is available at https://github.com/TangZJ/able-nerf.

Acknowledgements This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

References

[1] Michael Abrash. Quake's lighting model: Surface caching. *Graphic programming black book*, 2000. 2

[2] Relja Arandjelović and Andrew Zisserman. Nerf in detail: Learning to sample for view synthesis. *arXiv preprint arXiv:2106.05264*, 2021. 3

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF Inter-*

national Conference on Computer Vision, pages 5855–5864, 2021. 4, 5, 11

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 6

[5] Max Born and Emil Wolf. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013. 2

[6] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001. 3

[7] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996. 3

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 5

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 5

[10] Philip Dutre and Yves D Willems. Importance-driven monte carlo light tracing. In *Photorealistic Rendering Techniques*, pages 188–197. Springer, 1995. 2

[11] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 3

[12] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3

[13] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996. 3

[14] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 3

[15] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986. 2, 5

[16] Jaroslav Krivanek and Pascal Gautron. Practical global illumination with irradiance caching. *Synthesis lectures on computer graphics and animation*, 4(1):1–148, 2009. 2

[17] Samuli Laine and Tero Karras. Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1048–1059, 2010. 3

[18] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. 3

[19] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. *ACM Transactions on Graphics (TOG)*, 21(3):427–437, 2002. 2

[20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3, 4, 6

[21] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 3

[22] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 3

[23] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34:19313–19325, 2021. 3

[24] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 3

[25] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019. 3

[26] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8269–8279, 2022. 3

[27] Unity Technologies. Light probes. 2

[28] Unity Technologies. Reflection probe. 2

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5

[30] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 3, 4, 5, 6, 11

[31] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo

Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 3

[32] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 11

[33] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021. 11

[34] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 3

A. Optimisation Details

Our implementation is based on PyTorch Lightning. We optimise our model with Adam and we set the learning rate as 5×10^{-4} which is annealed log-linearly to 1×10^{-4}, with a warm-up phase of 1250 iterations. We set the hyper-parameters as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Each scene in the Blender and Shiny Blender dataset takes 1.5 days to train on 8 Tesla v100 GPUs with a total batch size of 8192 rays. To render an image on a single Tesla v100 GPU takes 110s.

B. Blender Dataset Details

We report full breakdown of individual scenes in the Blender Dataset in Tables A.1, A.2, A.3.

We compare our results with the rendered test set images from Ref-NeRF[1].

	chair	lego	materials	mic	hotdog	ficus	drums	ship
PhySG [33]	24.00	20.19	18.86	22.33	24.08	19.02	20.99	15.35
VolSDF [32]	30.57	29.46	29.13	30.53	35.11	22.91	20.43	25.51
Mip-NeRF [3]	35.12	35.92	30.62	36.76	37.34	33.19	25.36	30.52
Ref-NeRF [30]	35.83	36.25	35.41	36.76	37.72	33.91	25.79	30.28
Ours,No LE	35.76	36.62	34.57	35.90	38.68	34.28	25.98	30.60
Ours	36.25	38.03	35.46	37.11	39.07	35.69	26.84	31.75

A.1. Per-scene test set PSNRs on Blender Dataset. Results retrieved from [30].

	chair	lego	materials	mic	hotdog	ficus	drums	ship
PhySG [33]	0.898	0.821	0.838	0.933	0.912	0.873	0.884	0.727
VolSDF [32]	0.949	0.951	0.954	0.969	0.972	0.929	0.893	0.842
Mip-NeRF [3]	0.981	0.980	0.959	0.992	0.982	0.980	0.933	0.885
Ref-NeRF [30]	0.984	0.981	0.983	0.992	0.984	0.983	0.937	0.880
Ours, No LE	0.941	0.985	0.983	0.991	0.987	0.986	0.945	0.893
Ours	0.987	0.987	0.985	0.993	0.988	0.989	0.951	0.919

A.2. Per-scene test set SSIMs on Blender Dataset. Results retrieved from [30].

f	chair	lego	materials	mic	hotdog	ficus	drums	ship
PhySG [33]	0.093	0.172	0.142	0.082	0.117	0.112	0.113	0.322
VolSDF [32]	0.056	0.054	0.048	0.191	0.043	0.068	0.119	0.191
Mip-NeRF [3]	0.020	0.018	0.040	0.008	0.026	0.021	0.064	0.135
Ref-NeRF [30]	0.017	0.018	0.022	0.007	0.022	0.019	0.059	0.139
Ours, No LE	0.021	0.018	0.031	0.010	0.024	0.017	0.065	0.147
Ours	0.018	0.015	0.029	0.008	0.021	0.014	0.057	0.122

A.3. Per-scene test set LPIPS on Blender Dataset. Results retrieved from [30].

C. Shiny Blender Dataset Details

We report full breakdown of individual scenes in the Shiny Blender Dataset in Tables A.4, A.5, A.6.

We compare our results with the rendered test set images from Ref-NeRF.

[1]We thank Dor Verbin for providing us with the test images from Ref-NeRF model output

	teapot	toaster	car	ball	coffee	helmet
PhySG [33]	35.83	18.59	24.40	27.24	23.71	27.51
Mip-NeRF [3]	46.00	22.37	26.50	25.94	30.36	27.39
Ref-NeRF, no pred. normals [30]	47.09	23.32	27.19	26.09	31.79	30.54
Ref-NeRF [30]	47.90	25.70	30.82	47.46	34.21	29.68
Ours	47.30	26.52	28.76	36.62	33.01	31.04

A.4. Per-scene test set PSNRs on Shiny Blender Dataset. Results retrieved from [30].

	teapot	toaster	car	ball	coffee	helmet
PhySG [33]	0.990	0.805	0.910	0.947	0.922	0.953
Mip-NeRF [3]	0.997	0.891	0.922	0.935	0.966	0.939
Ref-NeRF, no pred. normals [30]	0.997	0.898	0.926	0.865	0.967	0.962
Ref-NeRF [30]	0.998	0.922	0.955	0.995	0.974	0.958
Ours	0.998	0.949	0.942	0.984	0.975	0.968

A.5. Per-scene test set SSIMs on Shiny Blender Dataset. Results retrieved from [30].

	teapot	toaster	car	ball	coffee	helmet
PhySG [33]	0.022	0.194	0.091	0.179	0.150	0.089
Mip-NeRF [3]	0.008	0.123	0.059	0.168	0.086	0.108
Ref-NeRF, no pred. normals [30]	0.006	0.134	0.064	0.272	0.087	0.068
Ref-NeRF [30]	0.004	0.095	0.041	0.059	0.078	0.075
Ours	0.007	0.092	0.052	0.107	0.114	0.080

A.6. Per-scene test set LPIPS on Shiny Blender Dataset. Results retrieved from [30].