NeX: Real-time View Synthesis with Neural Basis Expansion

Suttisak Wizadwongsa* Pakkapon Phongthawee* Jiraphon Yenphraphai*
Supasorn Suwajanakorn
VISTEC, Thailand
{suttisak.w.s19, pakkapon.p.s19, jiraphony.pro, supasorn.s}@vistec.ac.th

Abstract

We present NeX, a new approach to novel view synthesis based on enhancements of multiplane image (MPI) that can reproduce next-level view-dependent effects—in real time. Unlike traditional MPI that uses a set of simple RGBα planes, our technique models view-dependent effects by instead parameterizing each pixel as a linear combination of basis functions learned from a neural network. Moreover, we propose a hybrid implicit-explicit modeling strategy that improves upon fine detail and produces state-of-the-art results. Our method is evaluated on benchmark forward-facing datasets as well as our newly-introduced dataset designed to test the limit of view-dependent modeling with significantly more challenging effects such as the rainbow reflections on a CD. Our method achieves the best overall scores across all major metrics on these datasets with more than 1000× faster rendering time than the state of the art. For real-time demos, visit https://nex-mpi.github.io/

1. Introduction

Novel view synthesis is an exciting and long-standing problem that draws much attention from both the computer vision and graphics communities. The problem comprises two intriguing challenges of how to construct a visual scene representation from only a sparse set of images and how to render such a representation from unseen perspectives. A wide range of applications are possible from this area of research ranging from virtually visiting tourist attractions to viewing any online product all around in 3D; however, such experiences would only become most compelling and practical when the representation allows photo-realistic and real-time synthesis.

One candidate that can serve this purpose is multiplane image (MPI) [53] which approximates the scene’s light field with a set of parallel semi-transparent planes placed along a reference viewing frustum. This representation is shown to be more effective than traditional 3D mesh reconstruction in reproducing complex scenes with challenging occlusions, thin structures, or planar reflections. However, the standard RGBα representation of MPI is limited to diffuse surfaces whose appearance stays constant regardless of the viewing angle. This greatly limits the types of objects and scenes that MPI can capture. Recent research on implicit scene representation has made significant progress in the past months [21, 32, 18, 50, 42] and can be applied to view synthesis problem. Unfortunately, its expensive network inference still prohibits real-time rendering, and reproducing complex surface reflectance with high fidelity still remains a challenge. Our method breaks these limits on both fronts.

*Authors contributed equally to this work.
We introduce NeX, a new scene representation based on MPI that models view-dependent effects by performing basis expansion on the pixel representation in our MPI. In particular, rather than storing static color values as in traditional MPI, we represent each color as a function of the viewing angle and approximate this function using a linear combination of spherical basis functions learned from a neural network. Furthermore, we propose a hybrid parameter modeling strategy that models high-frequency detail in an explicit structure within an implicit MPI modeling framework. This strategy helps improve fine detail that is difficult to model by a neural network and produces sharper results in fewer training iterations.

We evaluate our algorithm on benchmark forward-facing datasets and compare against state-of-the-art approaches including NeRF [22] and DeepView [6]. These datasets, however, contain mostly diffuse scenes and fairly simple view-dependent effects and cannot be used to judge the new limit of our algorithm. Thus, we collect a new dataset, Shiny, with significantly more challenging view-dependent effects such as the rainbow reflections on a CD, refraction through non-planar glassware or a magnifying glass. Our method achieves the best overall scores across all major metrics on these datasets. We provide quantitative and qualitative results and ablation studies to justify our main technical contributions. Compared to the recent state of the art, NeRF [22], our method captures more accurate view-dependent effects and produces sharper results—all in real time.

2. Related Work

Learning MPIs. Multiplane image by Zhou et al. [53] is a scene representation that consists of parallel semi-transparent planes placed along a reference viewing frustum. Note that a similar representation has been proposed earlier by the name of “stack of acetates” by Szeliski & Golland [39]. Originally, MPI [53] is used to solve a small-baseline stereo problem and is inferred with a convolutional neural network (CNN) from an input stereo pair. Subsequent work extends MPI to support multiple input photos [6, 21, 17] or even infers an MPI from a single image [43]. In [21], a CNN is used to predict multiple nearby MPIs which are then blended together to produce the final output. Srinivasan et al. [37] predicts an MPI using a two-step process that combines 3D CNNs for MPI prediction and a 2D flow field for warping RGB values from an intermediate rendering. In contrast, DeepView [6] uses a CNN to learn gradient updates to the MPI instead of predicting it directly. This learned gradient descent helps avoid over-fitting and requires only a few iterations to generate an MPI. Recently, DeepMPI [17] has been introduced to model time-varying scene appearance and can manipulate colors on their MPI using a CNN. However, these approaches do not model view-dependent effects or only handle them indirectly by blending multiple view-independent MPIs. This greatly limits the types of applicable objects and scenes.

View synthesis and interpolation. One way to categorize view synthesis algorithms is by how dense the input scene is sampled. When the capture is dense as in lumigraph [9, 22] and light field rendering [16, 49, 15, 31], the challenge becomes how to store, interpolate, and compress the light field samples. When there are only 1-2 input images, the challenge becomes how to infer the ill-constrained 3D geometry and disoccluded regions [38, 23, 43, 47, 3]. Our work focuses on the case with a moderate number of captures facing forward. Besides MPI-based approaches, other solutions include methods based on layered depth images [29, 44, 4], Soft3D by Penner et al. [24], which combines depth estimation with soft blending of an estimated geometry, and other 3D reconstruction based techniques [54, 10].

Neural approaches to view synthesis include DeepStereo [7], which uses a CNN to predict pixels directly for individual viewing angles and Neural Textures [41], which combines a reconstructed 3D mesh with neural textures that can be rendered with a neural network. Similar ideas of using neural latent code stored in some geometric structure such as a voxel grid or volumes have been proposed [33, 5, 19]. Neural BTF [26] represents the bidirectional texture function with an encoder-decoder network that takes in the light and viewing angles and outputs each pixel’s color. [14] uses a generative adversarial network to model spatially varying BRDFs of specular microstructures.

One recent notable work is Neural Radiance Fields (NeRF) by Mildenhall et al. [22], which represents a 5D radiance field with a multilayer perceptron (MLP) that directly regresses the volume density and RGB colors. This method can handle view-dependent effects as the viewing angle is part of the 5D radiance function. Subsequent work improves upon NeRF by using explicit sparse voxel representation to improve fine detail (NVSF) [18], parameterizing the space to better support unbounded scenes (NeRF++) [50], incorporating learned 2D features that help enforce multiview consistency (GRF) [42], or extending NeRF to handle photometric variations and transient objects in internet photo collections (NeRF-W) [20]. Another related line of work involves implicitly modeling surface reflectance properties in addition to the scene geometry [1] or the light transport function [52]. Our work is inspired by these implicit neural representations as well as deep image prior [45], but our goal is directed toward a representation amendable to discretization and real-time rendering.

Light field factorization. Our reparameterization of pixel into a combination of basis functions is closely related to light field factorization approaches in many areas, such as surface light field [48], precomputed radiance transfer [36, 35], BRDF estimations [11], light field and tensor display [46]. In particular, our MPI pixel can be consid-
3.1. Original MPI Representation

Multiplane image [53] is a 3D scene representation that consists of a collection of $D$ planar images, each with dimension $H \times W \times 4$ where the last dimension contains RGB values and alpha transparency values. These planes are scaled and placed equidistantly either in the depth space (for bounded close-up objects) or inverse depth space (for scenes that extend out to infinity) along a reference viewing frustum (see Figure 2).

Rendering an RGB$\alpha$ MPI in any target view can be done by first warping all its planes to the target view via a homography that relates the reference and target view and apply the composite operator [25]. In particular, let $c_i \in \mathbb{R}^{H \times W \times 3}$ and $\alpha_i \in \mathbb{R}^{H \times W \times 1}$ be the RGB and alpha “images” of the $i$th plane, ordered from back to front. And denote $A = \{\alpha_1, \alpha_2, ..., \alpha_D\}$, $C = \{c_1, c_2, ..., c_D\}$ as the sets of these images. This MPI can then be rendered in a new view, $\hat{I}$, using the composite operator $O$:

$$\hat{I} = O(W(A), W(C))$$  

(1)

where $W$ is a homography warping function that warps each image to the target view, and $O$ has the form:

$$O(A, C) = \sum_{d=1}^{D} c_d T_d(A), \quad T_d(A) = \alpha_d \prod_{i=d+1}^{D} (1 - \alpha_i)$$  

(2)

This rendering equation is completely differentiable, thus allowing MPI to be inferred through image reconstruction loss [53, 6].

3.2. View-Dependent Pixel Representation

One main limitation of MPI is that it can only model diffuse or Lambertian surfaces, whose colors appear constant regardless of the viewing angle. In real-world scenes, many objects are non-Lambertian, such as a ceramic plate and a glass table. These objects exhibit view-dependent effects such as reflection and refraction. Reconstructing these objects with an MPI can make the objects appear unrealistically dull without reflections or even break down completely (Figure 6) due to the violation of the brightness constancy assumption used for matching invariant and 3D reconstruction. [21] attempts to solve this by combining multiple view-independent MPIs, but their results contain warping artifacts when blending between MPIs.

To allow for view-dependent modeling in our MPI, we modify the pixel color representation, originally stored as RGB values, by parameterizing each color value as a function of the viewing direction $\mathbf{v} = (v_x, v_y, v_z)$. This results in a 3-dimensional mapping function $C(\mathbf{v}) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ for every pixel. However, storing this mapping explicitly is

\* A single MPI can simulate planar reflections to some extent by placing the reflected content on one of its planes [6].
prohibitive and not generalizable to unobserved angles. Regressing the color directly from \(v\) (and the pixel location) with a neural network, as is done in e.g. [22], is possible though inefficient for real-time rendering. Our key idea is to approximate this function with a linear combination of learnable basis functions \(\{H_n(v) : \mathbb{R}^3 \rightarrow \mathbb{R}\}\) over the spherical domain described by vector \(v\):

\[
C^p(v) = k^p_0 + \sum_{n=1}^{N} k^p_n H_n(v)
\]  

(3)

where \(k^p_n \in \mathbb{R}^3\) for pixel \(p\) are RGB coefficients, or reflectance parameters, of \(N\) global basis functions. In general, there are several ways to define a suitable set of basis functions. Spherical harmonics basis is one common choice used heavily in computer graphics to model complex reflectance properties. Fourier’s basis or Taylor’s basis can also be used. However, one shortcoming of these “fixed” basis functions is that in order to capture high-frequency changes within a narrow viewing angle, such as sharp specular highlights, the number of required basis functions can be very high. This in turn requires more reflectance parameters which make both learning these parameters and rendering more difficult. With learnable basis functions, our modified MPI outperforms other versions with alternative basis functions that use the same number of coefficients shown in our experiment in Section 4.3.1.

To summarize, our modified MPI contains the following parameters per pixel: \(\alpha, k_0, k_1, \ldots, k_N\); and global basis functions \(H_1, H_2, \ldots, H_N\) shared across all pixels.

### 3.3. Modeling MPI with Neural Networks

Given our modified MPI and the differentiable rendering equation (Eq. 1), one can directly optimize for its parameters that best reproduce the training views. However, as demonstrated in earlier work [6], doing so would lead to a noisy MPI that overfits the training views and fails to generalize. We can overcome this problem by leveraging the idea of deep prior [45] and regressing these parameters with multilayer perceptrons (MLPs) from spatial conditioning, i.e., pixel coordinates. In other words, instead of allowing the estimated parameters to take arbitrary values which are prone to overfitting, we regularize these parameters by only allowing them to take on certain values that are in the span of a deep neural network’s output. In our case, we use two separate MLPs: one for predicting per-pixel parameters given the pixel location, and the other for predicting all global basis functions given the viewing angle. The motivation for using the second network is to ensure that the prediction of the basis functions, which are global, is not a function of the pixel location.

Our first MLP is modeled as \(F_\theta\) with parameter \(\theta\):

\[
F_\theta : (x) \rightarrow (\alpha, k_1, k_2, \ldots, k_N)
\]  

(4)

where \(x = (x, y, d)\) contains the location information of pixel \((x, y)\) at plane \(d\). Note that \(k_0\) is not predicted by \(F_\theta\) but will be stored explicitly in our implicit-explicit modeling strategy, explained in the upcoming section. The second network is modeled as \(G_\phi\) with parameter \(\phi\):

\[
G_\phi : (v) \rightarrow (H_1, H_2, \ldots, H_N)
\]  

(5)

where \(v\) is the normalized viewing direction.

It is interesting to note that in a study from [45], when a CNN is used as a deep prior for synthesizing images, the span of the CNN can capture the natural image manifold surprisingly well. In our case, we found that deep priors based on multilayer perceptrons can regularize our MPI and produce superior results compared to direct optimization without deep priors or with standard regularizers, such as total variation. In relation to NeRF [22], our MPI can be thought of as a discretized sampling of an implicit radiance field function that replaces the general view-dependent modeling, predicted with an MLP in NeRF, with more efficient basis functions.

### 3.4. Implicit-Explicit Modeling Strategy

One observation when using an MLP to model \(k_n\), or “coefficient images” when \(k^p_n\) is evaluated on all pixels \(p\), is the absence of fine detail (similar reports in [32, 22, 40]). In our problem, fine detail or high-frequency content tends to come from the surface texture itself and not necessarily from a complex scene geometry. Thus, we use positional encoding proposed in [22] to regress these images, which helps to an extent but still produces blurry results. Interestingly, we found that simply storing the first coefficient \(k_0\), or “base color,” explicitly helps ease the network’s burden of compressing and reproducing detail and leads to sharper results, also in fewer iterations. With this implicit-explicit modeling strategy, we predict every parameter with MLPs except \(k_0\) which will be optimized explicitly as a learnable parameter with a total variation regularizer.

**Coefficient Sharing:** In practice, computing and storing all \(N + 1\) coefficients for all pixels for all \(D\) planes can be expensive for both training and rendering. In our experiment, we use a coefficient sharing scheme where every \(M\) planes will share the same coefficients, but not the alphas. That is, there is a single set of \(\{K_0, \ldots, K_N\}\) for planes \(1\) to \(M\), and another set for planes \(M + 1\) to \(2M\), and so forth. With proper \(N\) and \(M > 1\), we do not observe any significant degradation in the visual quality, but a significant gain in speed and model compactness.

Finally, to optimize our model, we evaluate the two MLPs to obtain the implicit parameters, render an output image \(\hat{I}_i\), and compare it to the ground-truth image \(I_i\) from the same view. We use the following reconstruction loss:

\[
L_{\text{rec}}(\hat{I}_i, I_i) = \|\hat{I}_i - I_i\|^2 + \omega\|\nabla \hat{I}_i - \nabla I_i\|_1,
\]  

(6)
where $\nabla$ denotes the gradient operator and $\omega$ is a balancing weight [30]. Our approach is summarized in Algorithm 1.

**Algorithm 1: MPI training with NeX**

initialize: $\theta, \phi, K_0$;  
pre-compute $X$ pixel coordinate for each pixel;  
for iteration=0 to maxIter do  
    sampling image $I_i$;  
    compute $(A, \vec{K}) = F_{\theta}(X)$ where $\vec{K} = [K_1, K_2, \ldots, K_N]$; 
    compute viewing direction $V_i$ by $V_i = \mathbf{V}_i / \|V_i\|$;  
    compute view-dependent color $C = K_0 + \vec{K} \cdot H_\phi(V_i)$;  
    compute rendered image $I_i = O(W_i(A), W_i(C))$;  
    compute loss function by $L = L_o(I_i, I) + \gamma TV(K_0)$;  
    update $\theta, \phi, K_0$ with ADAM($\nabla_{\theta,\phi,K_0}L$);  
end  
Result: $A, K_0, K_1, \ldots, K_N$

3.5. Real-time Rendering

Every model parameter in our MPI can be converted to an image. This is done by evaluating $F_{\theta}$ on all pixel coordinates and $G_{\phi}$ on some pre-defined viewing span. Given these pre-computed images, we can implement Equation 1 in a fragment shader in OpenGL/WebGL and achieve real-time view-dependent rendering of our captured scenes.

4. Experiments

We perform quantitative and qualitative evaluations against state-of-the-art methods for novel view synthesis which include MPI-based methods and others. We also provide an extensive study on the choice of the basis functions and evaluate different variations of implicit-explicit modeling of the MPI parameters, ranging from fully implicit to fully explicit.

4.1. Implementation Details

Our model is optimized independently for each scene. The input photos are first calibrated and undistorted with a structure-from-motion algorithm from COLMAP [28]. In most of our experiments unless stated otherwise, we use an MPI with 192 layers with $M = 12$ consecutive planes sharing one set of texture coefficients.

**MLP architectures:** For $F_{\theta}$ that predicts per-pixel parameters given the pixel location $(x, y, d)$, we follow NeRF's [22] positional encoding and project input $x, y$ to 20 dimensions each and plane depth $d$ to 16 dimensions with the following projection $p(u) = [\sin(2^0 u), \cos(2^0 u), \ldots, \sin(2^5 u), \cos(2^5 u)]$ where input $u$ is first normalized to $[-1, 1]$. The total input dimension is 56. This network uses 6 fully-connected LeakyReLU layers, each with 384 hidden nodes. The output $\alpha$ uses a sigmoid activation, and the others use tanh activations. For $G_{\phi}$ that predicts the basis functions, we use positional encoding of the input view direction with 12 dimensions including 6 dimensions for $v_3$ and $v_4$. This network uses 3 fully-connected LeakyReLU layers with 64 hidden nodes to output 8 dimensions of $H_{\phi}(v)$.

**Training details:** To compute the loss, we randomly sample and render 8,000 pixels in the training view and compare them to the corresponding pixels in the ground-truth image. We set $\omega = 0.05$, $\gamma = 0.03$ and train our networks for 4,000 epochs using Adam optimizer [13] with a learning rate of 0.01 for base color and 0.001 for both networks and a decay factor of 0.1 every 1,333 epochs.

**Runtime:** For a scene with 17 input photos of resolution $1008 \times 756$, the training took around 18 hours using a single NVIDIA V100 with a batch size of 1. Our WebGL viewer can render this scene at 300 frames per second using an NVIDIA RTX 2080Ti. For comparison, NeRF took about 55 seconds to generate one frame on the same machine. In terms of FLOPs for rendering one pixel, we use 0.16 MFLOPs, whereas NeRF uses 226 MFLOPs.

4.2. Comparison to the State of the Art

We compare our algorithm to state-of-the-art MPI-based methods, DeepView [6] and LLFF [21], as well as non-MPI-based NeRF [22] and neural scene representations (SRN) [34]. We also compare to recent work, NSVF [18] in our supplementary material; however, their method focuses on object captures and is not designed to handle scenes with background due to the use of a bounded voxel grid. For evaluations, we use Spaces dataset from DeepView and Real Forward-Facing dataset from NeRF. Moreover, we introduce a significantly more challenging dataset, Shiny, to test the limit of view-dependent modeling.

4.2.1 Results on Real Forward-Facing Dataset

This dataset contains 8 scenes captured in real-world environments using a smartphone. The number of input images for each scene ranges from 20 to 62 images, each with a resolution of $1008 \times 756$ pixels. We use the same train/test split as NeRF and evaluate our test results using 3 metrics: PSNR (Peak Signal-to-Noise Ratio, higher is better), SSIM (Structural Similarity Index Measure, higher is better), and LPIPS [51] (Learned Perceptual Image Patch Similarity, lower is better).

As shown in Table 1, our method produces the highest average scores across all 3 metrics. We show scores for individual scenes in our supplementary. Note that we need to undistort the results from NeRF in order to match our
Table 1: Average scores across 8 scenes in Real Forward-Facing dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRN [34]</td>
<td>21.82</td>
<td>0.744</td>
<td>0.464</td>
</tr>
<tr>
<td>LLFF [21]</td>
<td>24.41</td>
<td>0.863</td>
<td>0.211</td>
</tr>
<tr>
<td>NeRF [22]</td>
<td>26.76</td>
<td>0.883</td>
<td>0.246</td>
</tr>
<tr>
<td>NeX (Ours)</td>
<td>27.26</td>
<td>0.904</td>
<td>0.178</td>
</tr>
</tbody>
</table>

Table 2: Average scores across 8 scenes in Shiny dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF [22]</td>
<td>25.60</td>
<td>0.851</td>
<td>0.259</td>
</tr>
<tr>
<td>NeX (Ours)</td>
<td>26.45</td>
<td>0.890</td>
<td>0.165</td>
</tr>
</tbody>
</table>

4.2.2 Results on Shiny Dataset

Our Shiny dataset also contains 8 scenes captured with a smartphone in a similar manner as Real Forward-Facing dataset. However, the scenes contain much more challenging view-dependent effects, such as the rainbow reflections on a CD, refraction through a liquid bottle or a magnifying glass, metallic and ceramic reflections, and sharp specular highlights on silverware, as well as detailed thin structures.

Table 2 shows that our method also outperforms NeRF on all 3 metrics on this dataset. In scene CD, our method can reproduce the rainbow reflections and the reflected image of a plastic cup on the CD, while NeRF fails to capture the reflected image, as seen in Figure 3. In scene Tools, our method captures a sharper image of the solder coil stand through the magnifying glass. In scene Food, our method captures the specular microgeometry of the textured ceramic plate with high fidelity. Our failure cases include the lack of sharp sparkles in the crystal candle holder in scene Food and the reflection of the tube rack in scene Lab shown in Figure 7. Currently, no other methods are able to handle extremely sharp highlights that only appear in one distinct location in each input view.

4.2.3 Results on DeepView’s Spaces Dataset

Spaces dataset contains indoor and outdoor captures using 16 forward-facing cameras on a fixed rig. Each image has a resolution of 800 × 480. We evaluate on the same 10 scenes in Spaces dataset as in DeepView. We train our model on 12 input views, then evaluate on 4 held-out views. Table 3 shows a comparison between Soft3D [24], DeepView [6], and our work. Note that DeepView only estimates an MPI with 80 planes, and these scores are computed from the test images released by those papers. Our method produces higher average scores than DeepView on all metrics for the 12-view setup. Figure 4 shows close-up results on one of the scenes from Spaces dataset. Note that DeepView focuses on sparser input setups than ours and can produce reasonable results with 4 input views by learning from a large dataset of scenes. However, it uses the original MPI representation which handles limited view-dependent effects.

4.3. Ablation Studies

We evaluate the effectiveness of our main contributions which are learned basis functions for view-dependent pixel representation and the implicit-explicit modeling strategy. For ablation studies, we train a 72-layer MPI with M = 6 sharing scheme and test on two scenes: “Tools,” which contains multiple types of view-dependent effects, and “Crest,” which contains high-detail patterns and thin structures from Shiny dataset. All images are in 1008 × 756 resolution.

4.3.1 View-dependent Modeling & Basis Functions

Number of basis coefficients: We vary the number of basis coefficients from zero, which represents no view-dependent modeling, to 20 and show quantitative results in Figure 5. The scores of our learned basis functions peak around 6-9 coefficients and show signs of overfitting afterward. Adding view-dependent modeling to MPI helps increase PSNR scores on all test scenes and significantly improves the visual quality for scenes with challenging lighting effects shown in Figure 6.

Types of basis functions: We compare our learned basis functions to other types of basis for modeling view-dependent effects by only changing \( H_n \) in Equation 3. We test the following basis options: Taylor Series (TS), Spherical Harmonics (SH), Hemispherical Harmonics (HSH) [3], Jacobi Spherical Harmonics (JH), and Fourier Series (FS).
Figure 3: Qualitative results on test views from NeRF’s real forward-facing dataset. Our method captures more complete geometry than LLFF and SRN in Orchids scene and recovers the most detail in Leaves scene.

Figure 4: The top row shows our rendered results. (a) Our method captures more accurate reflections on the table top. (b) Our method captures the reflected image of a plastic cup as well as the rainbow reflections, while NeRF produces a blurry and noisy result. (c) Our method produces a sharper image of the coil through the magnifying glass.

Spherical harmonics are commonly used for representing complex illumination and are derived from Legendre polynomials. However, since our captures are mostly forward-facing, the viewing directions from which a point on a surface can be observed will only span a hemisphere. Thus, we also evaluate alternative basis functions that are more suitable for this viewing span, namely Hemi-spherical harmonics [8], which are derived from shifted Legendre polynomials. Generalizing this further, one can derive modified spherical harmonics that target an even tighter viewing span than a hemisphere through shifted Jacobi polynomials (JH). The exact forms are shown in our supplementary material.

Figure 5 shows that our learned basis outperforms these fixed basis functions, even the ones whose viewing domains have been narrowed down, when the same number of coefficients is used. In principle, given a sufficiently expressive network, our learned basis can approximate other kinds of basis functions, if required, or reproduce higher frequencies using the same rank order, which is the number that dictates the highest frequency in those fixed basis functions.

4.3.2 Implicit Function and Explicit Structure

In this experiment, we validate our design decision that stores base color $K_0$ explicitly while modeling other pa-
parameters with implicit functions. Additionally, in Table 4 we explore all 8 alternatives for representing alpha (A), base color (K₀), and view-dependent coefficients (K₁, ..., Kₙ), ranging from fully implicit (Im-Im-Im) to fully explicit (Ex-Ex-Ex). Note that the fully explicit model corresponds to optimizing the rendering equation directly without any deep priors on the parameters. The result shows that our Im-Ex-Im outperforms other alternatives, and storing base color K₀ explicitly is beneficial also to other configurations regardless of the modeling choices for the alpha and coefficients. Qualitatively, our method produces significantly better detail compared to the fully implicit model and cleaner results that generalize better than the fully explicit model.

### 5. Limitations & Failure Cases

Our model is based on MPI and thus inherits similar limitations. When viewing our MPI from an angle too far away from the center, there will be “stack of cards” artifacts which can reveal individual MPI planes. Our model still cannot fully reproduce the hardest scenes in our Shiny dataset, which include effects such as light sparkles, extremely sharp highlights, or refraction through test tubes (Figure 7). Exposure differences in the training images that are not properly compensated may lead to flickering in the rendered output. Training our MPI still takes a long time and may require a higher number of input views to replicate view-dependent effects. Learning how to do this with fewer input images using a dataset of scenes or with learned optimizers [6] could be an interesting direction.

### 6. Conclusion

We have investigated a new approach to novel view synthesis using multiplane image (MPI) with neural basis expansion. Our representation is effective in capturing and reproducing complex view-dependent effects and efficient to compute on standard graphics hardware, thus allowing real-time rendering. Extensive studies on public datasets and our more challenging dataset demonstrate state-of-the-art quality of our approach. We believe neural basis expansion can be applied to the general problem of light field factorization and enable efficient rendering for other scene representations not limited to MPI. Our insight that some reflectance parameters and high-frequency texture can be optimized explicitly can also help recovering fine detail, a challenge faced by existing implicit neural representations.

### Acknowledgement

This research was supported by PTT public company limited, SCB public company limited, the Program Management Unit for Human Resources & Institutional Development, Research and Innovation Thailand (NXPO 1426293), and VISTEC. The authors also would like to thank Jeong Joon Park for useful discussions.
References


