Supplemental Materials

We strongly encourage the reader to view the video results available at https://nerf-w.github.io/.

A. Model Parameters

In the following, we document the selected hyperparameters for NeRF-W. A grid search over the hyperparameter space was employed to select the below values. Note that the values were optimized for the Brandenburg Gate scene of the Phototourism dataset and reused for the remaining scenes. As in NeRF, we employ an MLP architecture for modeling density and radiance. We apply 8 layers of 512 hidden units for the ‘Base’ component, and 4 layers of 128 hidden units for both ‘Static’ and ‘Transient / Uncertainty’ components; see Figure 3). Regarding our positional encoding functions γ we use 15 frequencies for when encoding position and 4 when encoding viewing direction.

All model embeddings are initialized using Xavier uniform initializer. In order to optimize the appearance embeddings, we freeze the weights of the network (in order to not alter the geometry of the 3D model) and only update the Generative Latent Appearance features to match the appearance of the requested image. This procedure is carried out every 1000 steps during training and the weights are optimized for 50 steps.

During training, 512 points per ray are sampled from each of the coarse and fine models for a total of 1024 points per ray. We double that number during evaluation to 2048. The latent embedding vector for appearance has an embedding dimension of size \( n^{(a)} = 48 \). For transient objects, we use an embedding dimension of size \( n^{(\tau)} = 16 \). We select a value of 0.01 for the \( L_1 \) regularizer multiplier \( \lambda_a \) and 0.03 as the minimum importance \( \beta_{\text{min}} \). Models are trained with Adam over 300,000 iterations with a batch size of 2048 and an initial learning rate of 0.001, decaying ten-fold every 150,000 iterations.

B. Evaluation

The problem of evaluating the performance of a view synthesis system on in-the-wild photo collections is itself a challenging problem, as it reduces to the still-open problem of measuring perceptual image similarity [26, 38, 40, 42]. Though LPIPS [42] is an acceptable solution in many contexts, two perfectly-aligned images of the same 3D structure imagined at different times of day or under different lighting conditions generally result in significant “perceptual” image differences. This presents a challenge when evaluating NeRF-W (and its ablation NeRF-A). After training, only images in the training set are assigned optimized appearance embeddings \( \ell^{(a)} \) — our model does not recover a single model of the world, it recovers a family of solutions under a variety of appearances. To evaluate a test-set image, we must therefore identify the image’s corresponding \( \ell^{(a)} \) embedding.

Naive solutions to this problem, such as setting \( \ell^{(a)} \) to a vector of zeros or to the mean of optimized embeddings, results in plausible renderings that, while structurally accurate, fail to match the appearance of the ground truth. As a result, perceptual metrics are unable to credibly measure the quality of each method’s scene representation. To address this deficiency, we evaluate NeRF-A and NeRF-W by optimizing appearance embedding \( \ell^{(a)} \) on the left half of each ground-truth image and calculating metrics on the corresponding right half. This split-image evaluation scheme enables NeRF-A and NeRF-W to adapt the scene’s appearance without directly optimizing held-out pixels. Note that by virtue of the model’s design, changes to the appearance embedding cannot alter the underlying scene’s geometry (see Figures 7 and 8), further limiting the potential for information leakage.

Finally, note that the NRW baseline captures appearance by encoding the entirety of a held-out image with an appearance encoder model. Unlike NeRF-W, this method is unable to isolate geometry from appearance and, given a sufficiently high-dimensional space, is capable of storing the image itself. In preliminary experiments, we notice a small drop in performance when applying a similar split-image evaluation scheme as described above. To remain comparable to prior results, we replicate the evaluation scheme as originally published.

For the LPIPS error metric we use the AlexNet implementation provided at https://github.com/richzhang/PerceptualSimilarity.
D. Lego Dataset

For a controlled ablation study, we construct variants of the Lego dataset [25] inspired by effects we expect to find in-the-wild (Table 3).

**Color perturbations:** To simulate variable lighting and exposure, we apply a random scale and shift transformation to the RGB values of each image. In particular, we replace each training image $I_i \in [0, 1]^{800 \times 800 \times 3}$ with $\tilde{I}_i$ where, for each RGB color channel $j$, $\tilde{I}_{ij} = \min(1, \max(0, s_{ij} I_{ij} + b_{ij}))$ where scale $s_{ij} \sim \mathcal{U}(0.8, 1.2)$ and offset $b_{ij} \sim \mathcal{U}(-0.2, 0.2)$ are sampled uniformly at random for each $i$ and $j$. Qualitatively, this results in variable tint and brightness. We apply perturbations to all training images except the first, whose appearance embedding is used to render novel views. The top row of Figure 12 shows the effect of applying random color perturbations to the same image.

**Occlusions:** We simulate transient occluders by drawing randomly-positioned and randomly-colored squares on each training image. Each square consists of ten vertical, colored stripes with colors chosen at random. Like transient occluders in the real world, these squares do not have a consistent 3D location from image to image. We again leave the first training image untouched for reference. Figure 12 shows the effect of adding occlusions randomly to the same view.

D.1. Experiments

For the Lego datasets, we optimize for 125,000 steps on 4 GPUs, which takes approximately 8 hours. We use the same NeRF hyperparameters reported by Mildenhall et al. [25] for all NeRF variants. We optimize models on 100 images and evaluate on an additional 200, using the same NeRF model hyperparameters presented in the original work. Hyperparameters specific to NeRF-W (those not present in NeRF) are tuned for each dataset variation via grid search.

**Original:** We begin by applying all methods to the original, unperturbed Lego dataset. Quantitatively, we find that all model variations perform similarly (Table 3). While NeRF achieves slightly higher PSNR than all NeRF-W variants, all other metrics suggest indistinguishable model quality. We find that our implementation of NeRF performs slightly better than the performance reported in the original NeRF paper [25].

**Color Perturbations:** We find that this change alone decreases NeRF’s PSNR by approximately 10dB on average (Table 3). As illustrated in Figure 13, NeRF is unable to isolate image-dependent photometric effects from its shared scene representation and thus entangles color variation with viewing direction. NeRF-A and NeRF-W, on the other hand, isolate tinting using the appearance embedding $\ell^{(a)}$. Novel

Table 2: Number of images and pixels per Phototourism scene. Pixel counts measuring in millions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Images</td>
<td>Pixels</td>
</tr>
<tr>
<td><strong>BRANDENBURG GATE</strong></td>
<td>763</td>
<td>564M</td>
</tr>
<tr>
<td><strong>SACRE COEUR</strong></td>
<td>830</td>
<td>605M</td>
</tr>
<tr>
<td><strong>TREVI FOUNTAIN</strong></td>
<td>1,689</td>
<td>1249M</td>
</tr>
<tr>
<td><strong>TAJ MAHAL</strong></td>
<td>811</td>
<td>581M</td>
</tr>
<tr>
<td><strong>PRAHUG</strong></td>
<td>2,000</td>
<td>1417M</td>
</tr>
<tr>
<td><strong>HAGIA SOPHIA</strong></td>
<td>606</td>
<td>434M</td>
</tr>
</tbody>
</table>

Figure 12: Examples of perturbations applied to the Lego dataset. The top row shows various color perturbations applied to the same view, whereas the bottom shows the effect of randomly adding occluders to the same view.
views rendered with a fixed appearance embedding demonstrate consistent color from all camera angles. Quantitatively, we find both methods maintain almost identical metrics to those achieved on the original dataset.

**Occluders:** As shown in Table 3, this variation reduces NeRF’s PSNR by 14dB on average. To reduce training error, NeRF and NeRF-A represent occluders as colored fog in 3D space, thereby causing the Lego figure to be obscured (Figure 13). While latent appearance embeddings were not designed to capture transient objects, we find that they enable NeRF-A to reduce error by learning a radiance field that imitates the color of the underlying 3D geometry. NeRF-A and NeRF-W are better able to isolate transient occluders from the static scene than their counterparts.

**Color Perturbations and Occluders:** When both color and occluder perturbations are simultaneously enabled, we observe a decrease in performance across all methods, with NeRF-W outperforming all baselines. We further observe significant variation in model accuracy for both NeRF and NeRF-U across five random seeds. Both methods are poorly equipped to cope with photometric effects and occasionally fail to model the scene at all.
Table 3: Quantitative evaluation of NeRF and our proposed extensions on the synthetic Lego dataset. We report mean ± standard deviation across 5 independent runs with different random initializations. Best and second best results are highlighted. On the ORIGINAL dataset, all models perform near identically. NeRF fails to varying degrees on the perturbed datasets because it has no mechanism to account for those perturbations. As expected, NeRF-U fails on COLORS, but improves over NeRF on OCCLUDERS. Likewise, NeRF-A performs well on COLORS but fails on OCCLUDERS. NeRF-W is the only model that handles both types of perturbations.
Figure 13: Example dataset perturbations and renderings from NeRF, NeRF-A, NeRF-U and NeRF-W. The leftmost column illustrates the perturbations that were applied to the training dataset but using the test image for comparison. All other columns show renderings from models trained on datasets with each corresponding perturbation. NeRF-A and NeRF-U are largely able to disentangle color and occluder perturbations in isolation while NeRF-W is able to do so simultaneously. Render by Blender Swap user Heinzelnisse / CC BY.
Figure 14: Further qualitative results from experiments on Phototourism dataset. NeRF-W is able to capture reflections (top row), consistent scene geometry at a distance (middle), and eliminate transient occluders (bottom). Photos by Flickr users yatani, jingjing, lricecsp / CC BY.