

Supplementary Materials for

Hyper-NeRF: Hyperspectral Neural Radiance Fields with Continuous Radiance and Transparency Spectra

Anonymous ICCV submission

Paper ID 12454

Contents

1. Introduction	1
2. Implementation Details	1
2.1. RGB Implementations	2
3. Training Details	2
3.1. Commentary on the Tools Scene	2
3.2. Loss Curves	3
4. Qualitative Example Results	3

1. Introduction

In this work, we demonstrated that Neural Radiance Fields (NeRFs) can be naturally extended to hyperspectral data and are a well-suited tool for hyperspectral 3D reconstruction. The implementation details provided in this supplemental document describe our simple approach to hyperspectral NeRF, but we anticipate future works by the community will improve upon our baseline implementation using our to-be-published dataset, future larger datasets, additional architecture and hyperparameter tuning, and recent advances in NeRFs.

Our full code will be made publicly available for the camera ready version.

2. Implementation Details

We openly admit that significant improvements could be made on our implementation, re-emphasizing that our primary contribution is having demonstrated that NeRFs with continuous wavelength representations can work well on hyperspectral data.

We build upon nerfstudio’s nerfacto implementation, from commit [ef9e00e](#). The original nerfacto pipeline and field are shown in Figs. 1 and 2 respectively.

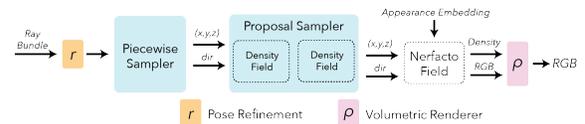


Figure 1. The original nerfacto pipeline (from [nerfstudio docs](#)) contains a proposal sampler, which is analogous to the “coarse” field from the original NeRF paper [2], and a “Nerfacto Field”, which is analogous to the primary network from the original NeRF paper (F_{Θ}).

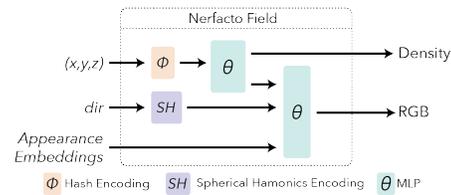


Figure 2. The original nerfacto field (from [nerfstudio docs](#)) is very similar to the original NeRF paper [2], but includes appearance embeddings [1] and uses slightly different encodings for the position and direction. This figure is reproduced in Fig. 2 of our main paper.

As briefly summarized in the main paper, we make minimal modifications to the pipeline and field. Using the notation from Section 5.4: Ablations, C_0 is the stock nerfacto field; C_1 only changes the rightmost MLP in Fig. 2 to output 128 channels in the last layer instead of 3; C_2 changes the positional hash encoding (ϕ in Fig. 2) to take 4 inputs instead of 3 (appending λ) and changes the rightmost MLP to only have 1 output for c^λ instead of (r, g, b) ; and C is shown in Fig. 2 (bottom) of the main paper. For C , the sinusoidal encoding for λ is taken to have 8 terms (tested 2, 4, 8, 16 terms, with 8 performing marginally better than 4 and 16, and 2 significantly worse). Also for C , the component $C(\lambda; \Theta_C)$ MLP from Fig. 2 of the main paper was taken to be identical to the rightmost MLP in Fig. 2 except

with the appropriate additional number of inputs to accommodate concatenating the sinusoidally encoded wavelength, and with only 1 output for c^λ instead of 3 for (r, g, b) . The latent vector Θ_C was taken to be the same size as in the nerfacto implementation (15-dim), with increasing the size to 32 and 64 showing negligible performance improvement but increased training instability.

Similarly, σ_0 is the stock nerfacto field; σ_1 only changes the left MLP in Fig. 2 to have 128 outputs; σ_2 changes the positional hash encoding to take 4 inputs, and σ is as shown in Fig. 2 (bottom) of the main paper. The additional component $\sigma(\lambda; \Theta_C)$ MLP has 3 layers with 64-dim hidden layers and ReLU activations. The sinusoidally encoded λ is shared with C and the latent Θ_σ vector is shared with (identical to) the Θ_C vector.

Finally, P_0 is the stock nerfacto proposal network while P_λ augments the proposal network with the wavelength. For P_λ , the position is first run through a hash encoding and MLP as in P_0 , except the MLP outputs a latent vector of dimension 7 instead of a scalar density. This latent vector is concatenated with a 2-term sinusoidally encoded wavelength and fed through a 2-layer network with 7-dim hidden layer to output a scalar density for inverse transform ray sampling. Like the original nerfacto pipeline, this sampling step occurs twice with identical architecture (but different weights) proposal networks.

Reiterating our implementation, our primary Hyper-NeRF implementation uses $C(\lambda; \Theta_C)$, $\sigma(\lambda; \Theta_\sigma)$, and P_0 , which we find to produce good results while also enabling wavelength interpolation.

2.1. RGB Implementations

Erratum. First, we apologize for the following error in the main paper that we will correct for the camera-ready version: For the caption in Table 1, we mistakenly state that our method outperforms the baseline for the more challenging Tools and Origami scenes. We intended to portray that our approaches achieve very comparable performance to standard RGB nerfacto on all scenes, despite the fact that, for Ours-Cont and Ours-RGB, the wavelength bands are very far apart which should make learning *more* difficult. For Ours-Hyper, we achieve comparable performance on all scenes except Tools despite the fact that we are learning 128 channels instead of just 3 while the number of learnable parameters is virtually identical to nerfacto and completely identical to Ours-Cont.

Pseudo-RGB wavelengths. For the purposes of generating pseudo-RGB images, we use the wavelengths 622nm, 555nm, and 503nm for R, G, and B channels respectively. Generating more accurate pseudo-RGB images by integrating over the spectrum according to an image sensor sensitivity curve (as described in Section 5.5 of the main paper)

would also be possible, but is unnecessary to demonstrate our results.

Hyper-NeRF RGB variation implementations. For the purposes of making a quantitative comparison to standard RGB NeRF, Section 5.2 and Table 1 of the main paper present variations of our approach applied to just 3-channel (RGB) images instead of the full 128-channel hyperspectral data. As described in the caption of Table 1, “Ours-Cont” refers to our Hyper-NeRF implementation but trained on only 3 wavelengths, “Ours-RGB” refers to C_1, σ_1, P_0 with 3 output channels for both C_1 and σ_1 , and “Ours-Hyper” refers to our Hyper-NeRF implementation trained on all 128 wavelengths. In the table for Ours-Hyper, PSNR and SSIM are evaluated over all 128 wavelengths while LPIPS is evaluated only for the 3 channels closest to the red, green, and blue wavelengths according to our Pseudo-RGB procedure.

3. Training Details

All networks were trained for 25000 steps, with 4096 train rays per batch using the Adam optimizer. The proposal networks and field both used lr=1e-2, eps=1e-15, and weight decay=1e-6. Camera extrinsic and intrinsic optimization were both turned off, since evaluation metrics are skewed if camera parameters are modified. To accommodate imperfect camera poses, after COLMAP, stock nerfacto was run on Pseudo-RGB images for 100000 steps with camera optimization turned on and the resulting camera pose corrections were saved and used in subsequent tests.

Of the 48 images per image set, 43 were used for training and 5 withheld for evaluation. Each step, the 4096 training rays were sampled randomly from all 43 training images, except for row 5 of the ablations where the training rays were sampled from only 10 of the 43 training images each step, with the choice of 10 images being re-sampled every 250 steps.

In some approaches, not all wavelengths could be run for every batch due to VRAM limits so a subset of wavelengths were sampled (randomly) for each batch, but every sampled wavelength was run for every ray in the batch. For rows 1 and 2 of the ablations, every wavelength could be run every batch. For rows 3, 4 (Hyper-NeRF, ours), and 5, the number of wavelengths sampled per step were 8, 12, and 6, respectively.

For evaluation, every wavelength of every pixel of the 5 evaluation images were evaluated and compared for each scene.

All tests were performed on an NVidia GeForce GTX 3090, and most training runs took between 20min-60min.

3.1. Commentary on the Tools Scene

The Tools scene experienced instabilities during training with several approaches including both Hyper-NeRF (ours)

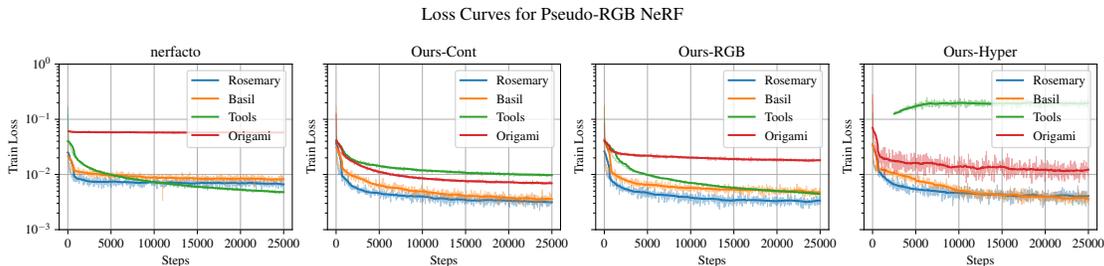


Figure 3. Loss curves for RGB NeRF correspond to the metrics from Table 1 in the main paper. Most scenes have converged by 25000 steps except the Tools scene which appears to have difficulty converging for all methods except “Ours-Cont”, which is reflected in Table 1 of the main paper.

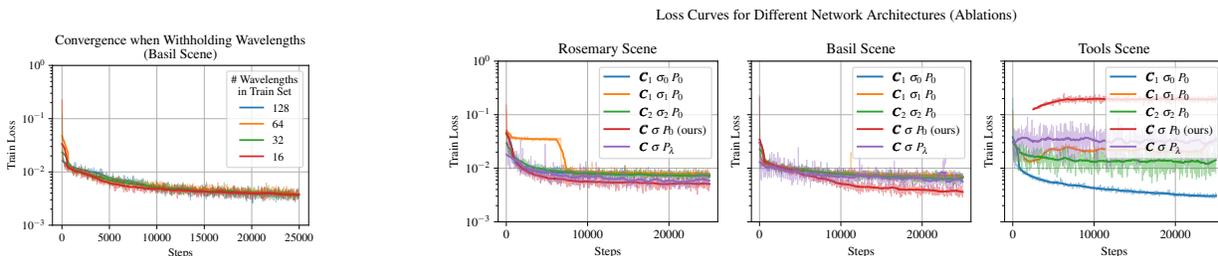


Figure 4. Loss curves for Hyper-NeRF trained with a subset of wavelengths (analogous to Table 2 in the main paper) shows that even training with only 1 out of every 8 wavelengths still has almost identical convergence rate w.r.t. number of steps.

Figure 5. Loss curves for ablation testing (analogous to Table 3 in the main paper) shows that while the rosemary and basil scenes optimize well, the tools scene does not converge particularly well for any method, re-emphasizing the suspected pre-processing (COLMAP) inaccuracy.

and nerfacto (RGB baseline). We anticipate that obtaining better camera intrinsics and extrinsics will correct this issue, since (a) every method had difficulty on this scene and (b) enabling camera pose optimization during NeRF training improved convergence for all methods. We plan to obtain better camera intrinsics by initializing COLMAP with the intrinsics obtained from other scenes, and we plan to obtain better camera extrinsics through a combination of tuning COLMAP parameters, utilizing turntable priors, and a longer NeRF-based camera pose refinement as described in 3. The poor convergence on the Tools scene for all methods is illustrated in both Fig. 3 (green curves) and Fig. 5.

3.2. Loss Curves

To demonstrate that all methods were fairly trained until convergence, the loss curves corresponding to the metrics given in the main paper are shown. As mentioned, the Tools scene appears to have difficulty converging for all methods including baseline nerfacto, suggesting possible pre-processing (COLMAP) inaccuracy. This is evident both in the green curves of Fig. 3 and in the rightmost plot of Fig. 5.

Exemplified by Fig. 5 (left, orange), one interesting observation we found is that the C_1 and σ_1 architectures occasionally exhibit convergence followed by a second descent and convergence. Inspecting the evaluation images, we ob-

serve the first convergence to be learning a scalar density field and the second convergence to be learning the color spectrum.

4. Qualitative Example Results

A selection of example images and videos are provided in both this pdf and in the enclosing zip folder to better gauge our results qualitatively. However, we emphasize again that our primary contribution is demonstrating that applying NeRF to hyperspectral data is a promising avenue for study. Therefore, we give 2 caveats: (1) visualizing pseudo-RGB results should not be compared to standard RGB NeRF results from other papers due to the many challenges associated with hyperspectral cameras as described in Section 4.2 of the main paper, and (2) results should be interpreted as a starting point for future works to iterate upon rather than a comprehensive approach.

In Figs. 6 and 7, we can observe that Hyper-NeRF visually appears “sharper” than all other approaches – the April-Tags clearly have much more detail in ours and the veins of the leaves also appear better resolved in Ours-RGB, Ours-Cont, and Ours-Hyper than the other approaches. These two figures depict pseudo-RGB representations of the hyperspectral images rendered by our NeRFs in subfigures (e)-(l). Meanwhile, (b)-(d) depict the RGB renderings of 3-

324 channel, RGB NeRFs. Figs. 6 and 7 also evidence that Ab-
325 lation 5 (wavelength-dependent sample proposal network)
326 is never able to learn colors despite having already con-
327 verged (Fig. 5). Finally, Figs. 6 and 7 also evidence that the
328 hyperspectral approaches do not always have perfect color
329 accuracy, tinting the AprilTags slightly green, which sug-
330 gests that increasing the size of the color network or latent
331 vector may produce better results.

332 Note that Figs. 6 and 7 are on the next page.
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431

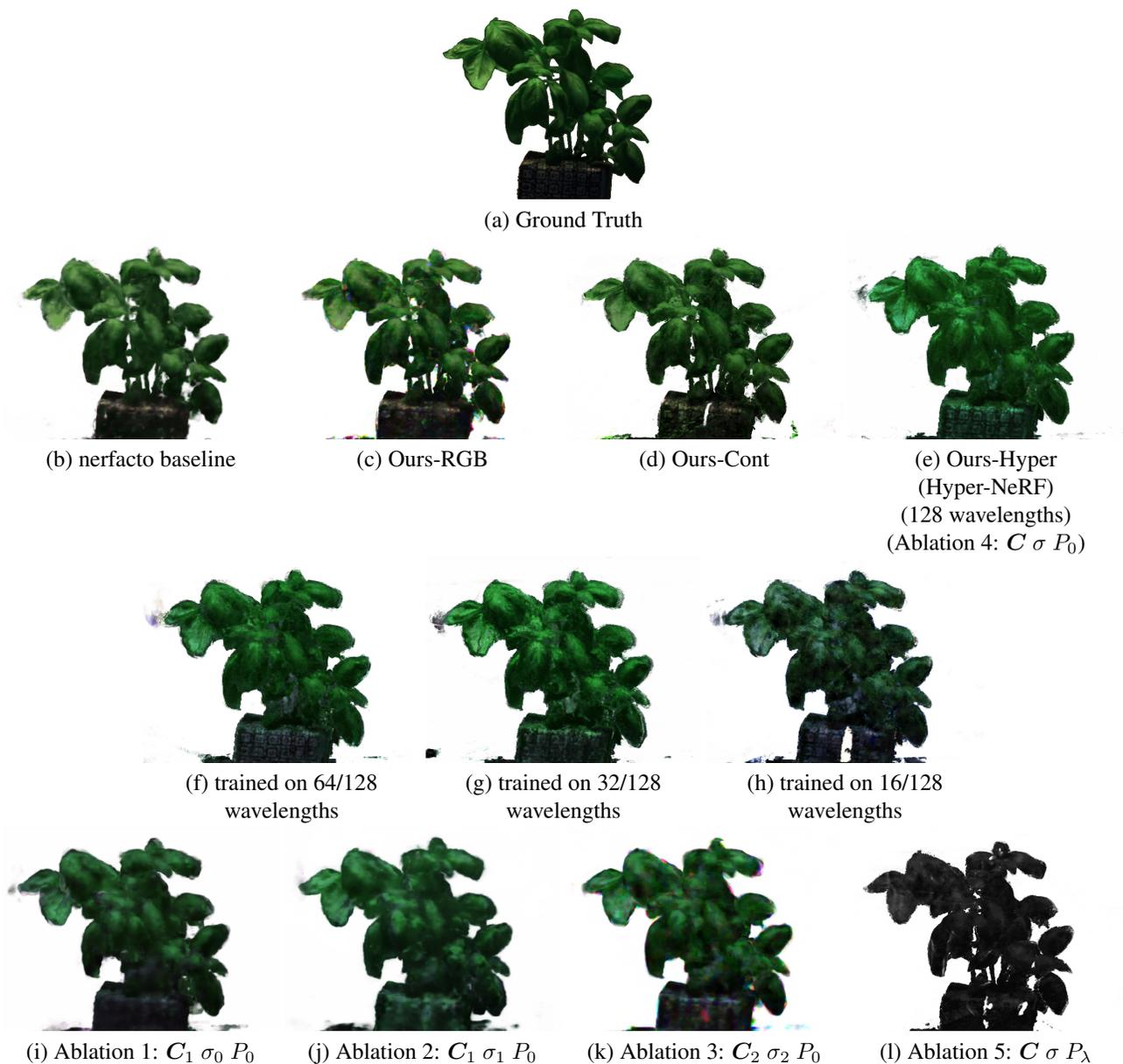


Figure 6. Pseudo-*RGB* images of an evaluation image from the Basil scene for different methods demonstrates that our approach (e) is able to capture more detail in the leaves and AprilTags than all other approaches including the nerfacto baseline. (b)-(e) represent the 4 approaches from Table 1 of the main paper and Fig. 3. (e)-(h) represent the 4 approaches from Table 2 of the main paper and Fig. 4. (e) and (i)-(l) represent the 5 ablations from Table 3 of the main paper and Fig. 5. While the NeRF representations for (b)-(d) contain only *RGB*, all other subfigures are pseudo-*RGB* representations of the hyperspectral NeRFs.

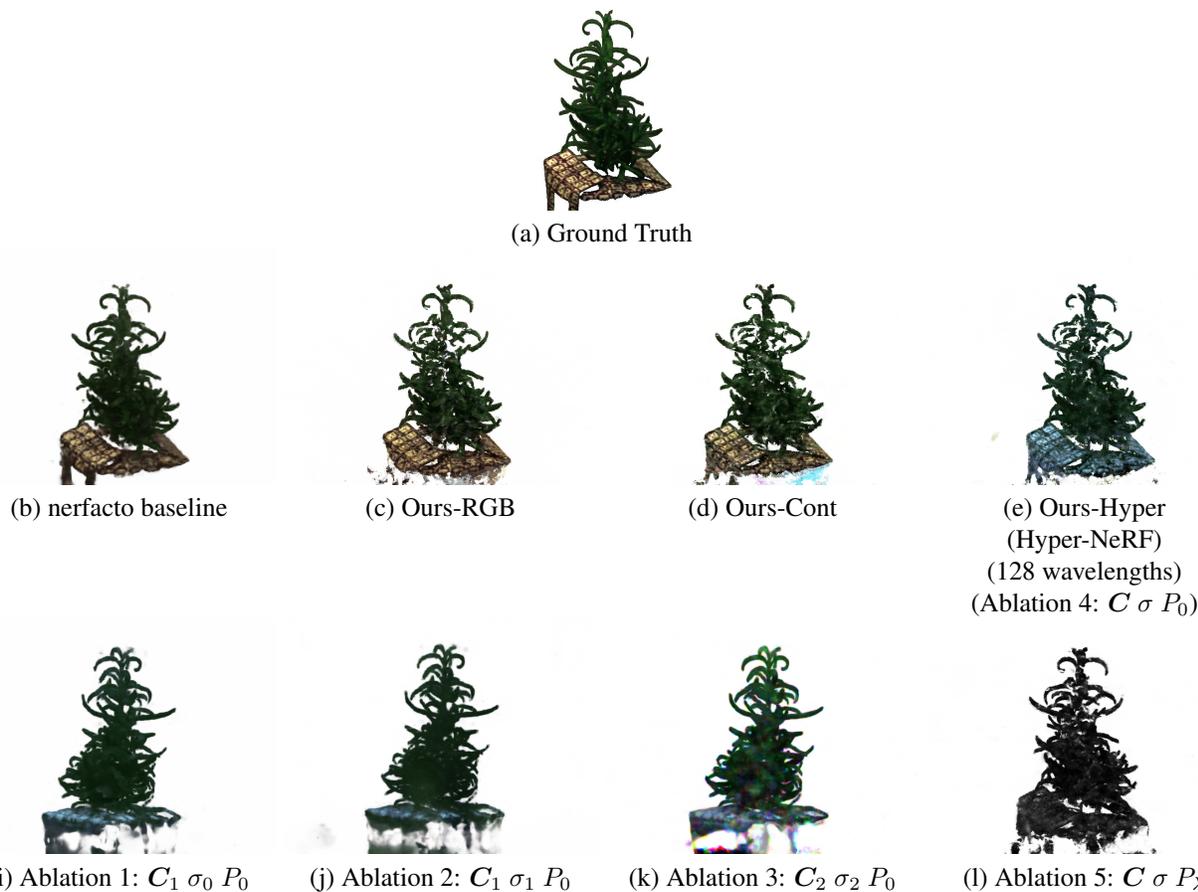


Figure 7. Same as Fig. 6 but for the Rosemary scene. Again, our approach appears to generate sharper results in the leaves and AprilTags than all other approaches including the nerfacto baseline. Contrary to the Basil scene, however, Ours-RGB and Ours-Cont also appear to generate results that have comparable sharpness to Hyper-NeRF.

648 **References**

- 649 [1] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi,
650 Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duck-
651 worth. NeRF in the Wild: Neural Radiance Fields for Uncon-
652 strained Photo Collections. In *CVPR*, 2021. 1
- 653 [2] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik,
654 Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf:
655 Representing scenes as neural radiance fields for view synthe-
656 sis. In *ECCV*, 2020. 1

702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755