Supplementary Material for AR-NeRF: Unsupervised Learning of Depth and Defocus Effects from Natural Images with Aperture Rendering Neural Radiance Fields

Takuhiro Kaneko

NTT Communication Science Laboratories, NTT Corporation

Appendix

This appendix provides detailed analyses (Appendix A), additional qualitative results (Appendix B), and implementation details (Appendix C).

A. Detailed analyses

In this section, we present seven detailed analyses to provide a deeper understanding of the proposed model.

- Appendix A.1: Importance of learning defocus effects.
- Appendix A.2: Effect of aperture ray sampling scheme.
- Appendix A.3: Simultaneous control of viewpoint and defocus.
- Appendix A.4: Generation of higher-resolution images.
- Appendix A.5: Application to defocus renderer.
- Appendix A.6: Fréchet inception distance.
- Appendix A.7: Gradient of difference in depth.

A.1. Importance of learning defocus effects

As discussed in Section 4.2, we represent aperture rendering in a ray-tracing framework, which is the basis of NeRF. Thus, our idea is general; its application is not restricted to AR-NeRF and can be incorporated into any NeRF-based model, including existing or pretrained models. This fact raises the question of whether it is necessary to learn the defocus effect from the data because we can attach the defocus effects virtually by incorporating aperture rendering into the model, even if aperture rendering is not used in the training.

In the main text, we present evidence verifying the importance of learning defocus effects from some perspectives. More specifically, the results in Table 2 indicate that the learning of the defocus effects is useful for improving model performance in terms of both image quality and depth accuracy. In this section, we examine the tolerance to defocus manipulation.

Oxford Flowers	σ_s	$2\sigma_s$	$3\sigma_s$	$4\sigma_s$	$5\sigma_s$
pi-GAN++	8.27	10.63	19.40	32.04	43.94
AR-NeRF	7.86	9.07	14.21	22.09	31.91
CUB-200-2011	σ_s	$2\sigma_s$	$3\sigma_s$	$4\sigma_s$	$5\sigma_s$
pi-GAN++	9.90	11.15	14.26	19.95	29.66
AR-NeRF	6.81	7.26	9.43	13.26	19.77
FFHQ	σ_s	$2\sigma_s$	$3\sigma_s$	$4\sigma_s$	$5\sigma_s$
pi-GAN++	4.64	6.94	11.54	17.10	23.81
AR-NeRF	3.66	5.30	9.51	14.98	21.60

Table 3. **Changes in the KID↓ ($\times 10^3$) when varying the aperture size to $s \in \{\sigma_s, 2\sigma_s, 3\sigma_s, 4\sigma_s, 5\sigma_s\}$.** σ_s is the standard deviation of the aperture size obtained by training AR-NeRF.

When a model can manipulate the defocus strength to the greatest extent possible, we believe that it can generate an image that is close to a real image even when large defocus effects are imposed. Based on this consideration, we examined the KID when large defocus effects were imposed on pi-GAN++ (i.e., a model without learning the defocus effects) and AR-NeRF (i.e., a model that learns the defocus effects). More precisely, we calculated the KID when the aperture size $s \in \{\sigma_s, 2\sigma_s, 3\sigma_s, 4\sigma_s, 5\sigma_s\}$, where σ_s is the standard deviation of the aperture size obtained through AR-NeRF training.

Results. We summarize the results in Table 3. We found that for every dataset and aperture size, AR-NeRF outperformed pi-GAN++ in terms of the KID score. In particular, we found that the difference is significant in the Oxford Flowers and CUB-200-2011 datasets, where viewpoint cues are limited or difficult to obtain and defocus cues play a critical role in obtaining a better SIDE (as shown in Table 2). These results indicate that (1) neural radiance fields need to be optimized for defocus effects to obtain defocus-tolerant representations that can be jointly used in various ranges of defocus strength, and (2) AR-NeRF (i.e., a model that learns the defocus effects) is useful for addressing this problem, particularly when other cues (e.g., viewpoint cues) are limited.

A.2. Effect of aperture ray sampling scheme

As discussed in Section 4.4, stratified sampling [17] was used to represent the aperture using a limited number of rays. We used five rays in particular: the origin of one ray was placed at the center of the aperture, and the origins of the others were placed along the circumference of the aperture at equal intervals. In this section, we examine the effect of this approximation.

Increasing the number of rays in the *training* phase is costly; thus, we examined the effect of an aperture ray sampling scheme in the *inference* phase. More specifically, we compared *stratified sampling*, which was used in the main experiments, with *random sampling*, where the offsets of ray origins, that is, $\mathbf{u}$ in Equation 3, were randomly sampled in a disk of radius s (i.e., $|\mathbf{u}| \in [0, s]$). To examine the effects of the number of rays, we investigated the difference in performance when changing the number of rays in $\{1, 2, 5, 10, 20\}$. As mentioned above, we examined the difference in performance in an inference phase. Hence, the base-trained model was the same as that used in Section 5 and was common across all settings.

Results. We summarize the results in Table 4. We found that, when we use random sampling, (1) the performance is improved as the number of rays increases until reaching approximately 10,[1] (2) the model with five rays (in the fourth column) performs worse than the model with the same number of rays with stratified sampling (in the last column), and (3) we need to increase the number of rays to 10 (in the fifth column) to obtain a performance comparable to that of the model with stratified sampling (in the last column). Considering that processing time and memory increase as the number of rays increases, we believe that stratified sampling with five rays is a reasonable choice in our experimental settings.

It is a possible that we will need to use more rays when applying to higher-resolution images, and in that case, we will need to increase the number of points along the ray.

A.3. Simultaneous control of viewpoint and defocus

AR-NeRF is a natural extension of NeRF, in which pinhole camera-based ray tracing is replaced with aperture camera-based ray tracing. This extension does not contradict the basic functionalities of NeRF. Thus, AR-NeRF can learn viewpoint-aware representations by taking over the characteristics of NeRF.

[1]The reason why the performance degrades when the number of rays increases too much (particularly in the cases where the number of rays is 20 on CUB-200-2011 and FFHQ) is that, in training, NeRF (including AR-NeRF) is optimized using *finite* sample points. Consequently, *overly dense* sampling in the inference phase can cause discrepancies from optimized conditions. This may lead to degradation when the number of rays increases too much. In other experiments, we found that the same phenomenon occurs when the number of sample points along the ray increases significantly.

Oxford Flowers	1	2	5	10	20	(5)
AR-NeRF	18.80	12.17	8.51	7.65	**7.37**	(7.86)
CUB-200-2011	1	2	5	10	20	(5)
AR-NeRF	13.25	8.92	6.97	**6.72**	7.19	(6.81)
FFHQ	1	2	5	10	20	(5)
AR-NeRF	15.92	8.26	4.26	**3.79**	4.08	(3.67)

Table 4. **Changes in the KID↓ ($\times 10^3$) when the number of rays varies within $\{1, 2, 5, 10, 20\}$.** When calculating the scores from the second to sixth columns, we randomly sampled the offsets of ray origins, that is, $\mathbf{u}$ (Equation 3), in a disk of radius s (i.e., $|\mathbf{u}| \in [0, s]$). Bold font indicates the best scores. When calculating the scores in the last column, we used stratified sampling, as detailed in Section 4.4. To distinguish them, we used parentheses for the latter.

Results. We demonstrate this strength in Figure 5. These results indicate that by using AR-NeRF, we can manipulate viewpoints and defocus effects simultaneously and independently in a unified framework.

A.4. Generation of higher-resolution images

In the experiments in the main text (Section 5), we used 64×64 images to examine various cases efficiently, following an AR-GAN study [8]. However, as discussed in Appendix A.3, AR-NeRF is a natural extension of NeRF; therefore, it can be applied to higher-resolution images with an increase in calculation cost, similar to other generative variants of NeRF, such as [1, 22]. To validate this statement, we applied AR-NeRF to the 128×128 images.

Results. We provide the examples of generated images in Figure 6. Following the experimental settings in the pi-GAN study [1], we trained the model using 128×128 images and rendered the final results by sampling 512×512 pixels. We found that AR-NeRF can render the defocus effects, including changes in defocus strength and focus distance, reasonably well, even in higher-resolution images.

A.5. Application to defocus renderer

After training, AR-NeRF can generate images from randomly sampled latent codes while varying the defocus strength and focus distance with photometric constraints. By utilizing these images, we can train a defocus renderer, which, given an image, manipulates the defocus strength and focus distance intuitively and continuously. We call this renderer *AR-NeRF-R*. In particular, we implemented AR-NeRF-R using a conditional extension of U-Net [21, 30], which incorporates the aperture size s and focus distance f as auxiliary information to control the image generation based on them. We provide the implementation details in Appendix C.2. In this section, we empirically investigate the effectiveness of AR-NeRF-R.

Dataset. We used the Oxford Flowers dataset to train AR-NeRF and AR-NeRF-generated images to train AR-NeRF-R. In particular, for AR-NeRF, we used the model discussed in Section 5. When training AR-NeRF-R, we used 128×128 images generated by AR-NeRF, where we increased the resolution of the generated images from 64×64 to 128×128 by increasing the density of the input points (Section 5.2), to allow AR-NeRF-R to be applied to 128×128 images. To confirm the generality of the learned model, we evaluated it on a different dataset (*iPhone2DSLR Flower* [29]), including photographs of flowers taken by smartphones. We provide the details regarding the dataset in Appendix C.2.

Comparison model. To the best of our knowledge, no previous method can learn the continuous representations of defocus strength and focus distance from natural images in the same setting as our own (i.e., *without* any supervision and any predefined model). Therefore, we used two baselines that have partially the same objective. The first baseline is *CycleGAN* [29], which trains a defocus renderer using *set level* supervision.[2] In contrast to AR-NeRF-R, CycleGAN requires additional supervision to determine whether each training image is an all-in-focus or focused image. The second baseline is *AR-GAN-DR* [8], which can train a defocus renderer without any supervision or pretrained model, similar to AR-NeRF-R; however, its conversion is one-to-one, and it cannot adjust the defocus strength and focus distance continuously.

Results. We present examples of the rendered images in Figure 7. We found that CycleGAN often performs unnecessary changes (e.g., color changes in the fifth row), whereas AR-NeRF-R and AR-GAN-DR do not. We infer that the aperture-rendering mechanisms in AR-NeRF and AR-GAN contributed to this phenomenon. The difference between AR-GAN-DR and AR-NeRF-R is that in AR-GAN-DR, the defocus strength and focus distance are uniquely determined according to the input image, whereas in AR-NeRF-R, we can change them continuously by varying the auxiliary information of aperture size s and focus distance f. This new AR-NeRF-R functionality allows for interactive selection of defocused images.

A.6. Fréchet inception distance

In the main text, we used KID because it has an unbiased estimator and complements the flaws of other representative metrics (i.e., Fréchet inception distance (FID) [7] and inception score (IS) [7]). However, the FID is a widely used metric. For reference, we report the FID in Tables 5 and 6. As also shown in a previous study [12], we found that KID and FID had high correlations in this case. These results do not contradict the statements in the main text.

[2]We used the pretrained model provided by the authors: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

	Oxford Flowers		CUB-200-2011		FFHQ	
	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓
AR-GAN	20.4	11.23	24.0	14.30	10.4	5.75
AR-GAN++	19.0	10.18	23.0	13.91	9.9	5.43
RGBD-GAN	20.8	12.04	24.6	14.92	11.6	6.73
AR-NeRF	17.1	7.86	17.0	6.81	7.8	3.67

Table 5. **Comparison of FID↓ and KID↓ ($\times 10^3$) between baseline GANs and AR-NeRF (ours).** This table supplements Table 1.

	(B)	(D)	(V)	Oxford Flowers FID↓	Oxford Flowers KID↓	CUB-200-2011 FID↓	CUB-200-2011 KID↓	FFHQ FID↓	FFHQ KID↓
pi-GAN			L	12.6	3.69	14.8	5.04	9.7	4.29
pi-GAN++	✓		L	18.2	8.30	21.5	9.84	8.6	4.43
AR-NeRF-0	✓	✓	0	15.2	6.81	20.1	8.67	8.0	3.83
AR-NeRF-F	✓	✓	F	–	–	–	–	8.8	4.59
pi-GAN++-F	✓		F	–	–	–	–	9.8	5.06
AR-NeRF	✓	✓	L	17.1	7.86	17.0	6.81	7.8	3.67

Table 6. **Comparison of FID↓ and KID↓ ($\times 10^3$) between AR-NeRF and ablated models.** This table supplements Table 2. Check marks (B) and (D) indicate the use of a background synthesis network and defocus cue, respectively. In column (V), L, F, and 0 indicate the use of local, full, and no viewpoint changes, respectively.

	Oxford Flowers			CUB-200-2011			FFHQ		
	KID↓	SIDE↓	∇d↓	KID↓	SIDE↓	∇d↓	KID↓	SIDE↓	∇d↓
AR-GAN	11.23	4.46	6.94	14.30	3.58	4.99	5.75	4.21	5.73
AR-GAN++	10.18	4.42	7.01	13.91	3.61	4.99	5.43	4.88	7.37
AR-NeRF	**7.86**	**3.94**	**3.54**	**6.81**	*3.63*	**3.39**	**3.67**	**2.61**	**2.24**

Table 7. **Comparison of KID↓ ($\times 10^3$), SIDE↓ ($\times 10^2$), and ∇d↓ ($\times 10^2$) among AR-GAN, AR-GAN++, and AR-NeRF (ours).** This table supplements Table 1.

A.7. Gradient of difference in depth

Figures 13–15 show that AR-GAN/AR-GAN++ yields unexpected artifacts around the edge and surface; however, SIDE can ignore this degradation because it measures the difference based on l_2, causing statistical averaging. This may explain why the improvement in depth prediction by AR-NeRF is not reflected in SIDE on the CUB-200-2011 dataset (Table 1), despite the qualitative difference (Figure 14). To validate this hypothesis, we calculated the gradient of the difference between the ground truth and predicted depths (∇d), which is commonly used to examine local structural similarity [4]. Table 7 lists the results and shows that AR-NeRF can improve depth prediction even on the CUB-200-2011 dataset in this metric.

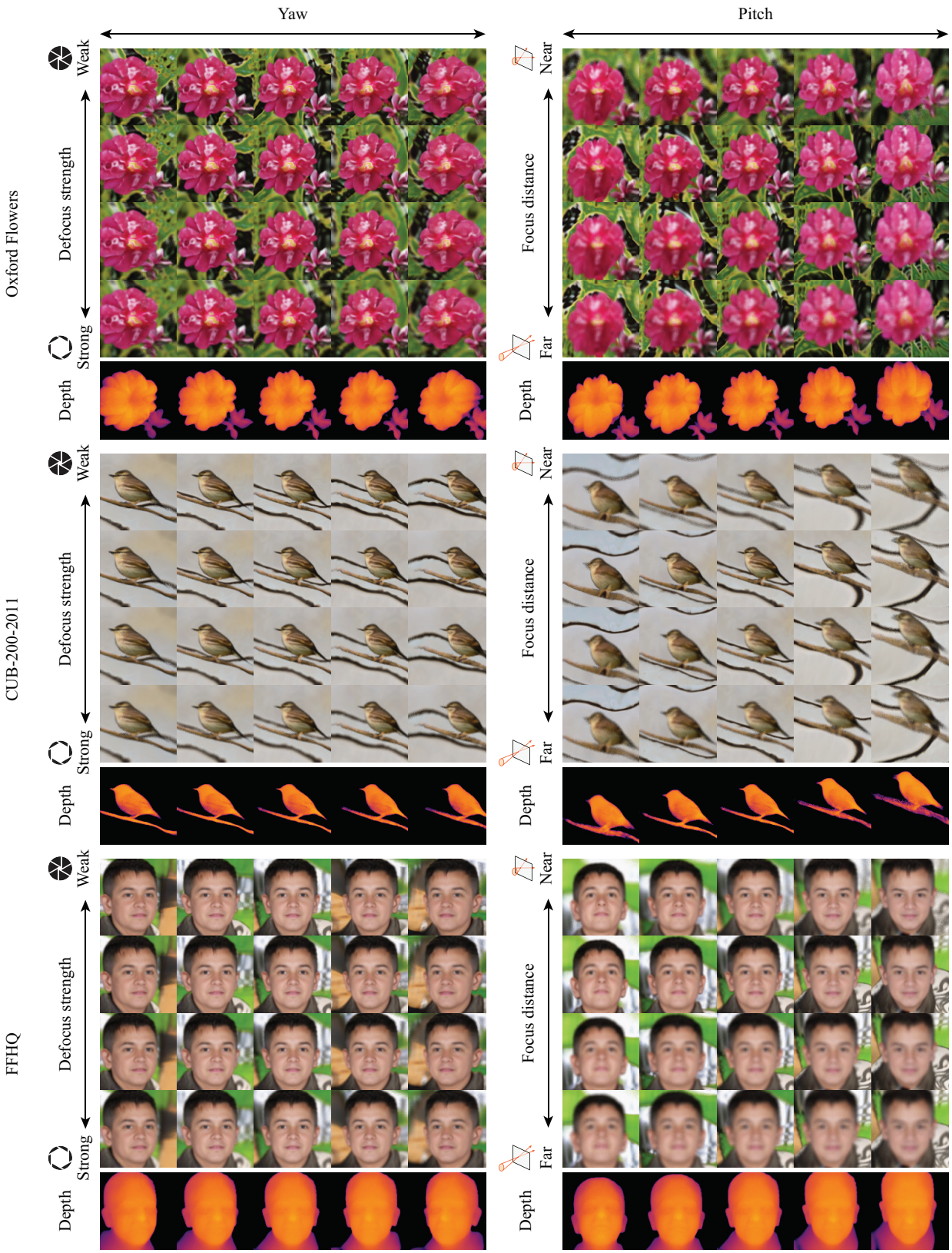


Figure 5. **Simultaneous control of viewpoint and defocus.**

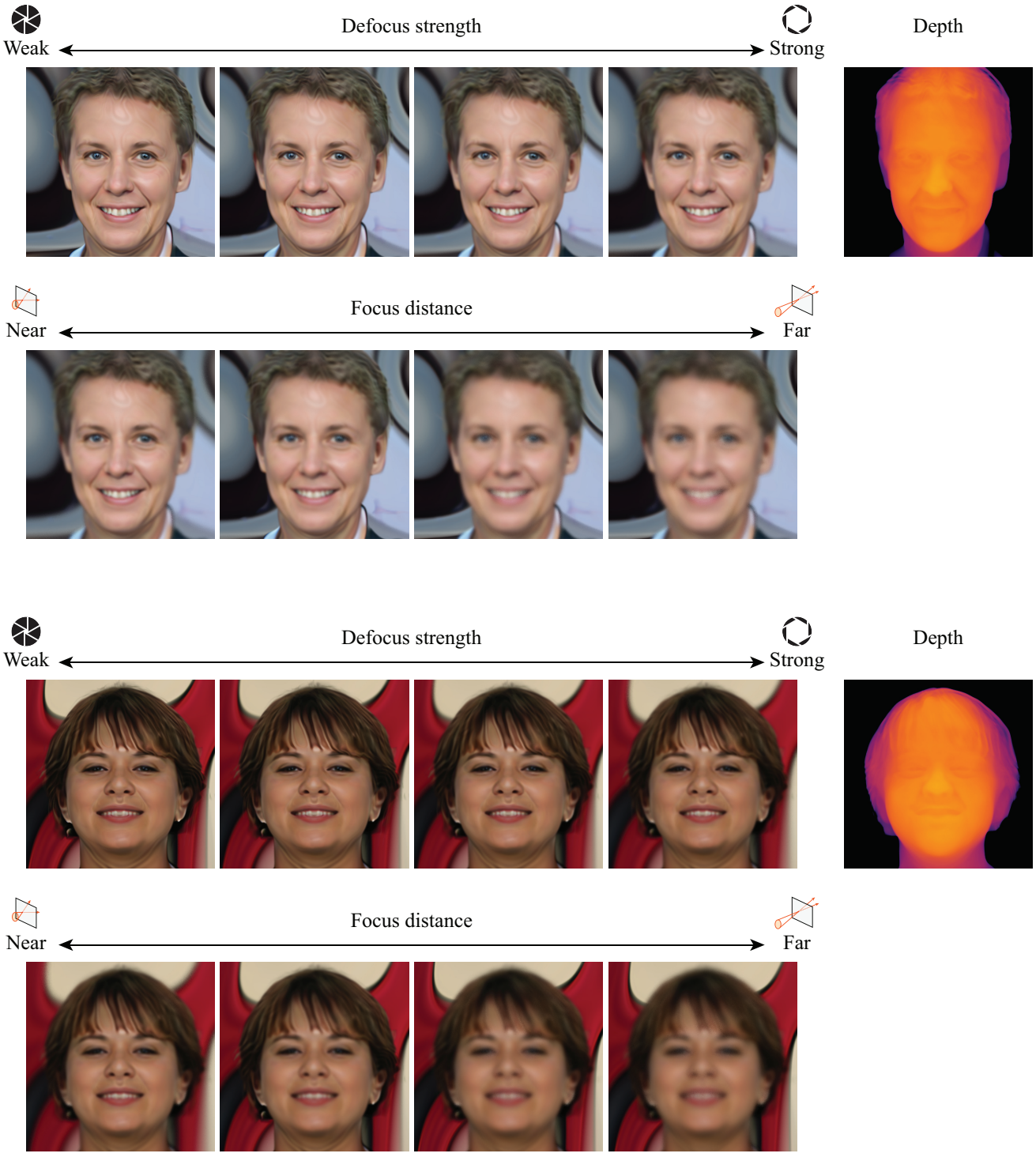
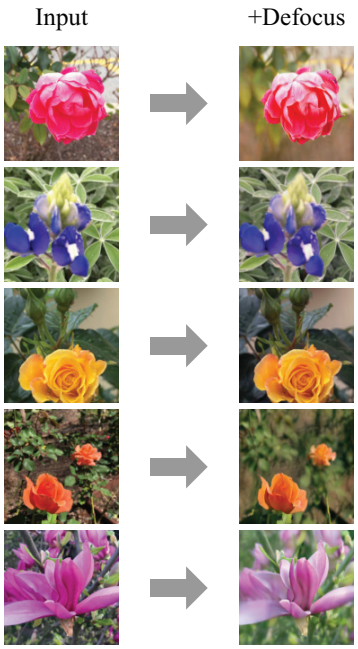
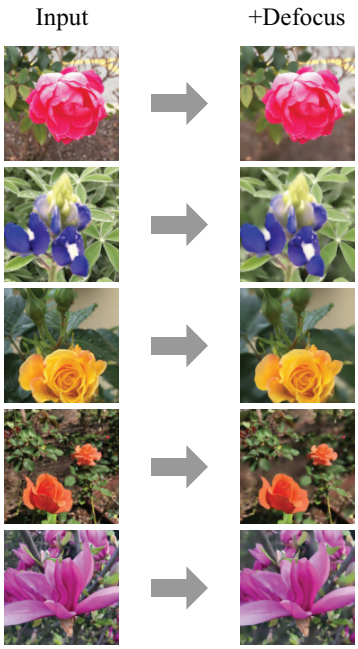


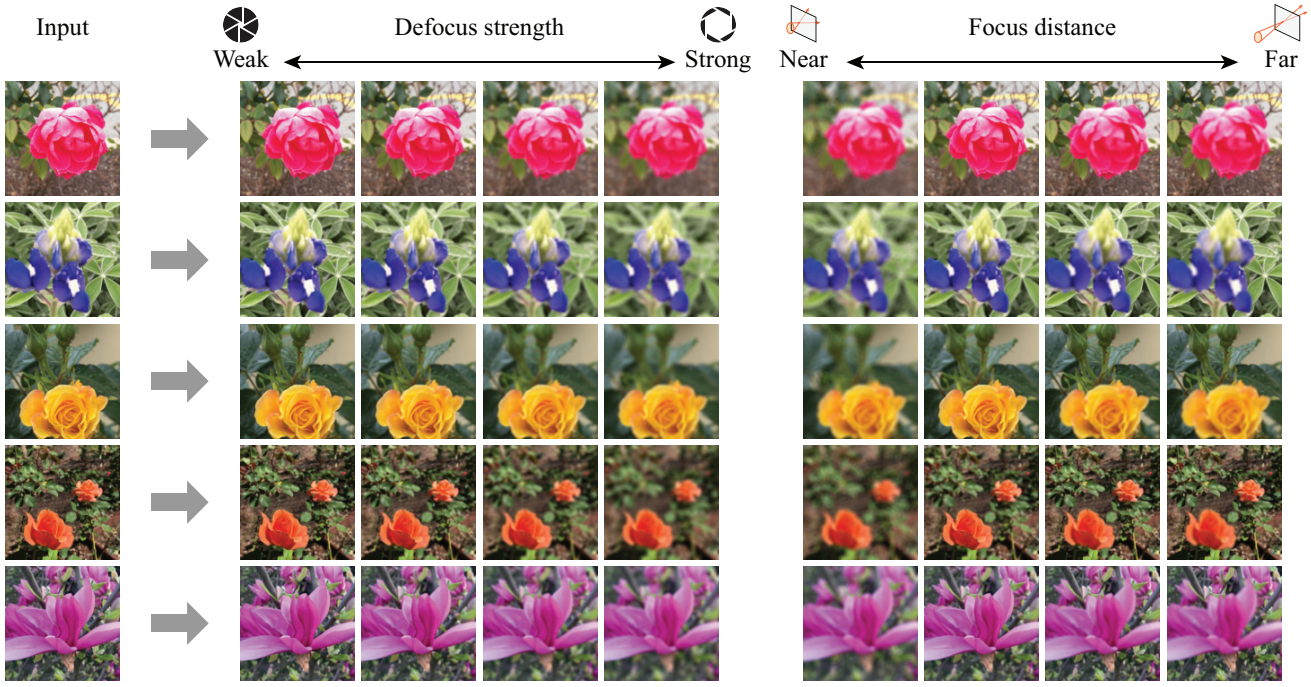
Figure 6. **Examples of** 512×512 **image generation using AR-NeRF.**



(a) CycleGAN



(b) AR-GAN-DR



(c) AR-NeRF-R (ours)

Figure 7. **Comparison of defocus rendering among CycleGAN, AR-GAN-DR, and AR-NeRF-R (ours).**

B. Additional qualitative results

In this appendix, we provide additional qualitative results that correspond to those presented in the main text. Figure captions and their relationship to the results in the main text are as follows:

- Figure 8: Unsupervised learning of depth and defocus effects from unstructured (and view-limited) natural images. This figure is an extended version of Figure 1.
- Figure 9: Simultaneous control of defocus and latent codes on the Oxford Flowers dataset. This figure is an extension of Figure 1.
- Figure 10: Simultaneous control of defocus and latent codes on the CUB-200-2011 dataset. This figure extends Figure 1.
- Figure 11: Simultaneous control of defocus and latent codes on the FFHQ dataset. This figure is an extended version of Figure 1.
- Figure 12: Comparison of generated images and depths between AR-GAN++ and AR-NeRF. This figure extends Figure 3.
- Figure 13: Comparison of depth prediction on the Oxford Flowers dataset. The depths are used to calculate the SIDEs in Tables 1 and 2.
- Figure 14: Comparison of depth prediction on the CUB-200-2011 dataset. The depths are used to calculate the SIDEs in Tables 1 and 2.
- Figure 15: Comparison of depth prediction on the FFHQ dataset. The depths are used to calculate the SIDEs in Tables 1 and 2.

Figure 8. **Unsupervised learning of depth and defocus effects from unstructured (and view-limited) natural images.** This figure is an extended version of Figure 1. As shown here, our objective is to acquire a generator that can generate sets of images and depths using only a collection of unstructured single images and without any supervision (e.g., ground-truth depth, pairs of multiview images, defocus supervision, and pretrained models). In particular, in the generation of an image, we aim to obtain a generator that can intuitively and continuously adjust the defocus strength and focus distance with photometric constraints.

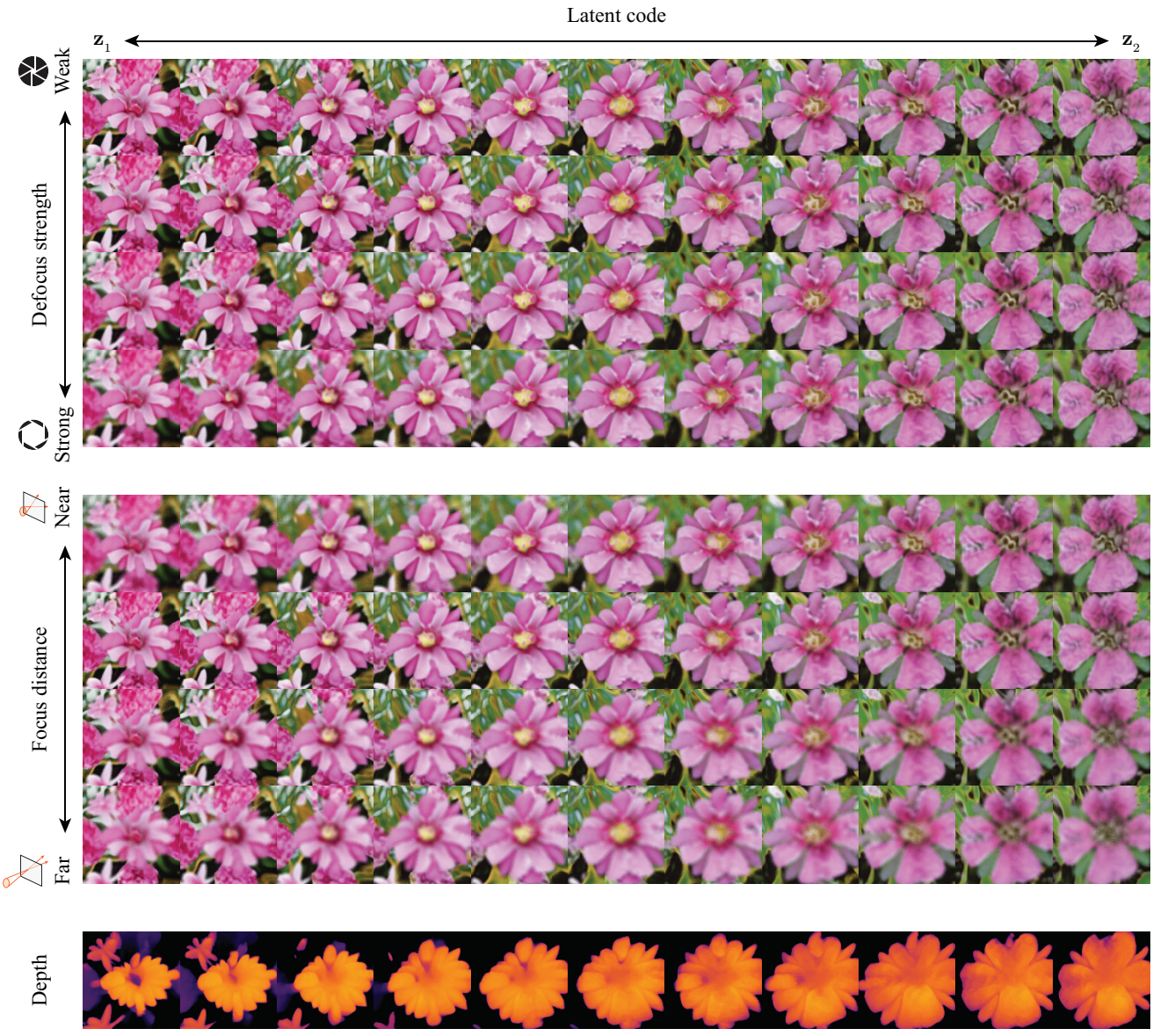


Figure 9. **Simultaneous control of defocus and latent codes on the Oxford Flowers dataset.** This figure is an extended version of Figure 1. In AR-NeRF, aperture randomized training (Section 4.3) encourages the defocus effects and latent codes to capture independent representations. By employing this characteristic, we can manipulate defocus effects and latent codes independently and simultaneously.

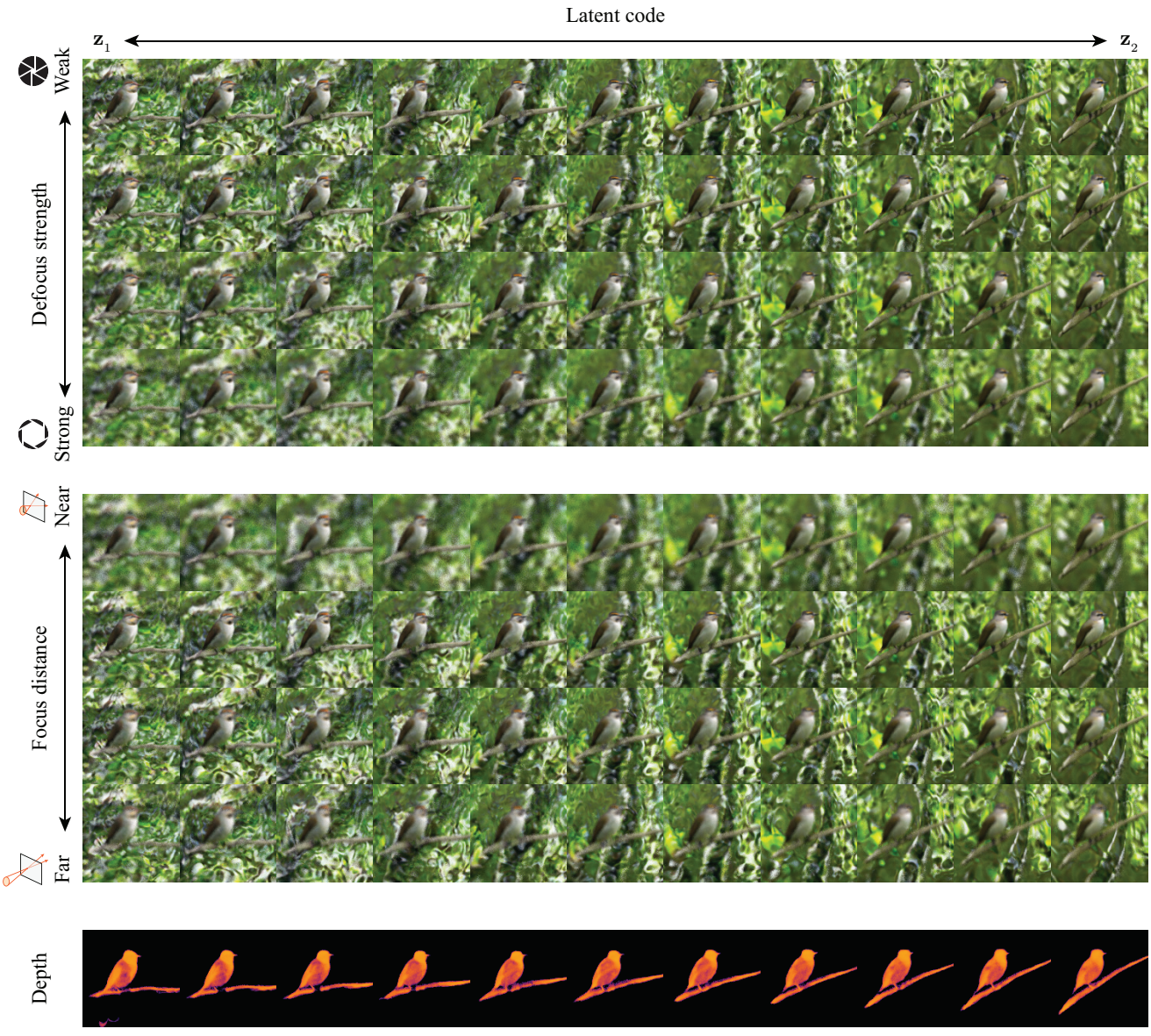


Figure 10. **Simultaneous control of defocus and latent codes on the CUB-200-2011 dataset.** This figure is an extended version of Figure 1. In AR-NeRF, aperture randomized training (Section 4.3) encourages the defocus effects and latent codes to capture independent representations. By employing this characteristic, we can manipulate defocus effects and latent codes independently and simultaneously.

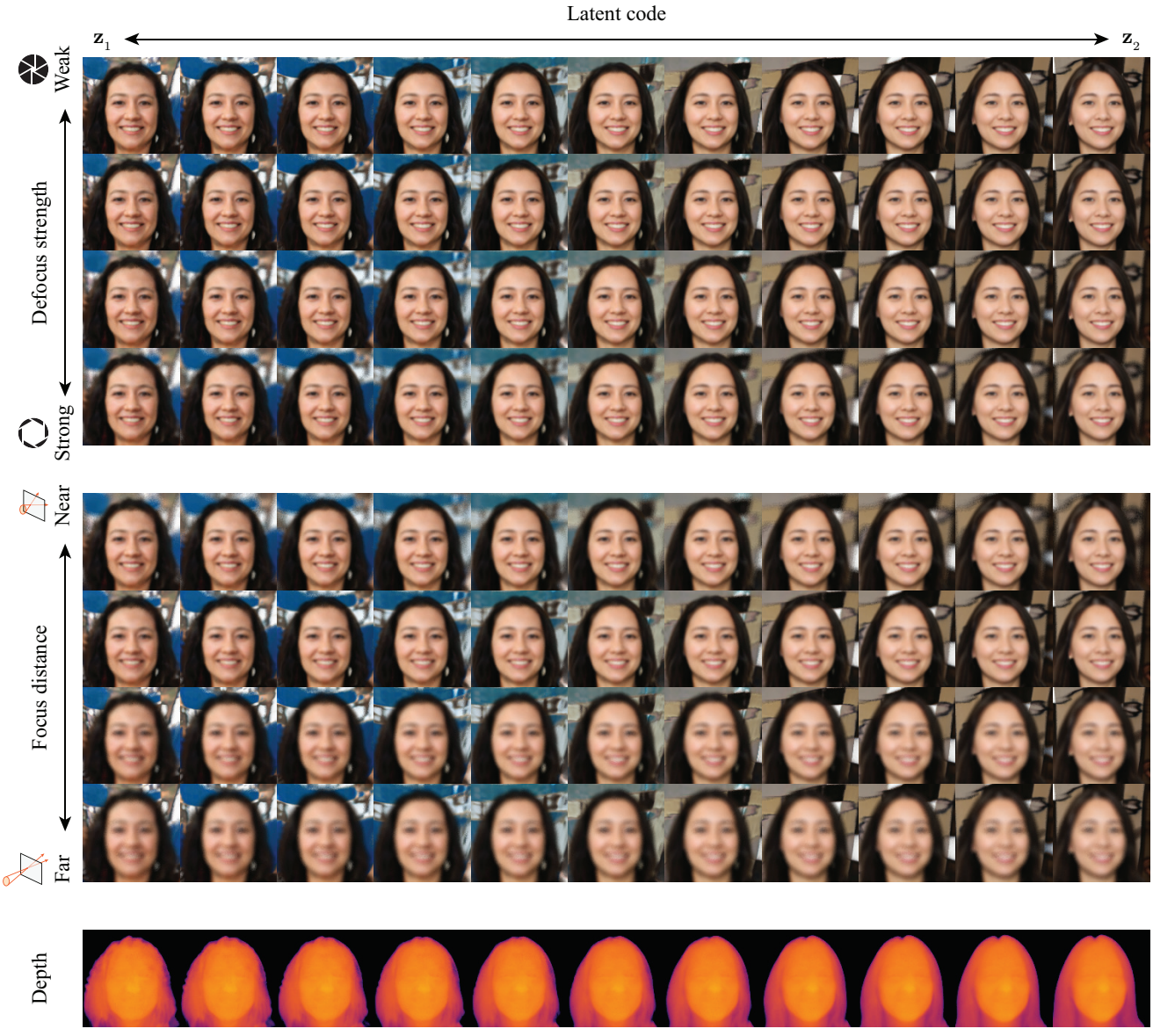


Figure 11. **Simultaneous control of defocus and latent codes on the FFHQ dataset.** This figure is an extended version of Figure 1. In AR-NeRF, aperture randomized training (Section 4.3) encourages the defocus effects and latent codes to capture independent representations. By employing this characteristic, we can manipulate defocus effects and latent codes independently and simultaneously.

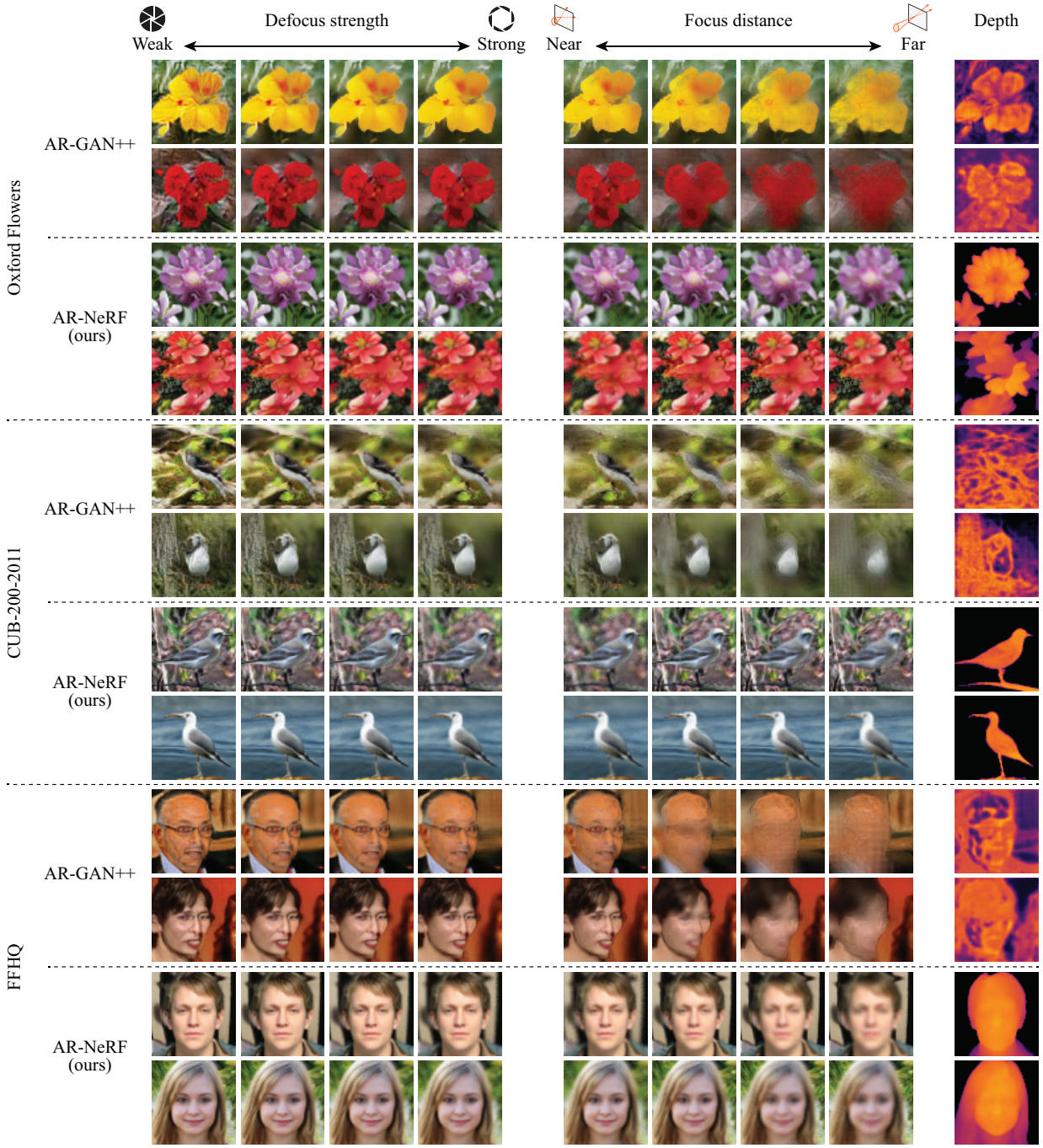


Figure 12. **Comparison of generated images and depths between AR-GAN++ and AR-NeRF (ours).** This figure is an extended version of Figure 3. We found that AR-NeRF can manipulate both the defocus strength and focus distance without producing significant artifacts. In particular, it is worth noting that AR-NeRF can refocus on both the foreground (shown in the fifth column) and background (shown in the second-to-last column), which are almost the same as those in the all-in-focus images (shown in the first column), by manipulating the focus distance. By contrast, AR-GAN++ tends to yield unexpected artifacts (e.g., over-smoothing or discretization artifacts), particularly when there is a strong defocus (shown in the fourth column) or refocus on the background (shown in the second-to-last column). As discussed in the main text, the possible causes for these phenomena are: (1) AR-GAN++ discretely represents light fields in a 2D space; thus, the discretization error becomes critical when a large manipulation is performed, and (2) the predicted depths (shown in the last column) contain artifacts (e.g., holes emerge in the objects), resulting in errors when images are rendered based on the depths. The properties of AR-NeRF, that is, (1) continuous representation in a 3D space and (2) joint optimization using defocus and viewpoint cues, are useful for addressing these weaknesses.

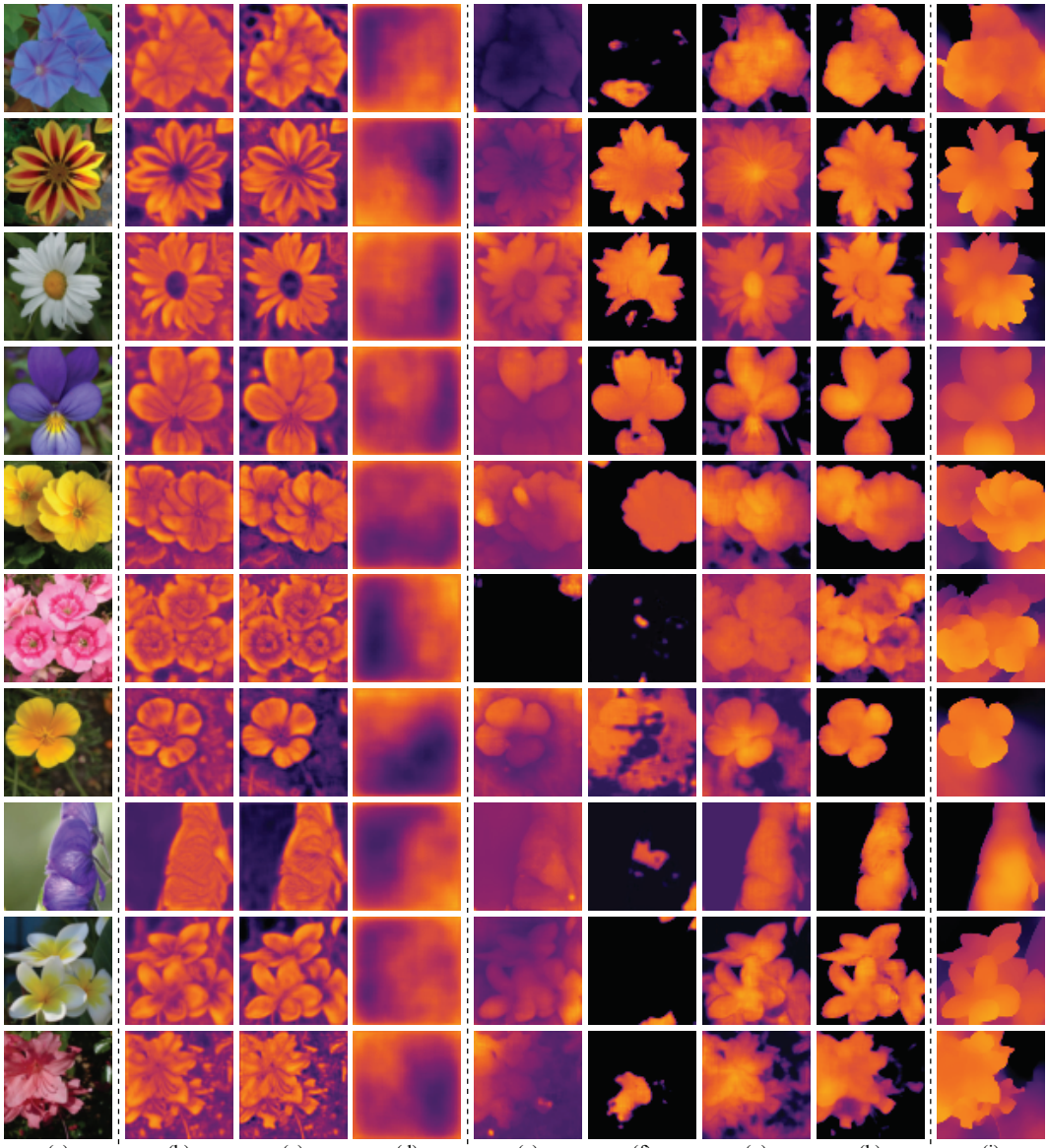


Figure 13. **Comparison of depth prediction on the Oxford Flowers dataset.** These depths are used to calculate the SIDEs in Tables 1 and 2. AR-GAN (b), AR-GAN++ (c), and RGBD-GAN (d) are CNN-based and are trained in a *fully* unsupervised manner. In addition, pi-GAN (e), pi-GAN++ (f), AR-NeRF-0 (g), and AR-NeRF (h) are NeRF-based and trained in a *fully* unsupervised manner. By contrast, the model in (i) [26] was trained using *paired* supervision and applied as the *ground truth* in the evaluation. The SIDEs in Tables 1 and 2 were calculated by comparing the depths in (b)–(h) with the depths in (i). Our findings are summarized as follows: (1) Because AR-GAN (b) and AR-GAN++ (c) only employ defocus (appearance) cues, their predicted depths are affected by their appearance. For example, in the second row, the pattern in the petals affects depth prediction, despite its non-necessity. (2) RGBD-GAN (d) utilizes viewpoint (geometric) cues for 3D representation learning. However, there are few viewpoint cues in this dataset; consequently, this model has difficulty in learning depth. (3) For the same reason, pi-GAN (e) and pi-GAN++ (f), which only employ viewpoint cues, suffer from learning difficulties, although NeRF itself has a strong 3D consistency at the design level. In particular, they fail to consistently distinguish between the foreground (flower) and background (surroundings), parts of which are often missing. (4) Although the results of AR-NeRF-0 (g) are closest to those of AR-NeRF (h), AR-NeRF-0 (g) is often affected by the appearance (e.g., in the second row, similar to AR-GAN (b) and AR-GAN++ (c)) because it also leverages only focus cues. (5) AR-NeRF (h) overcomes the limitations of pi-GAN (e), pi-GAN++ (f), and AR-NeRF-0 (g) by utilizing both the viewpoint and defocus cues.

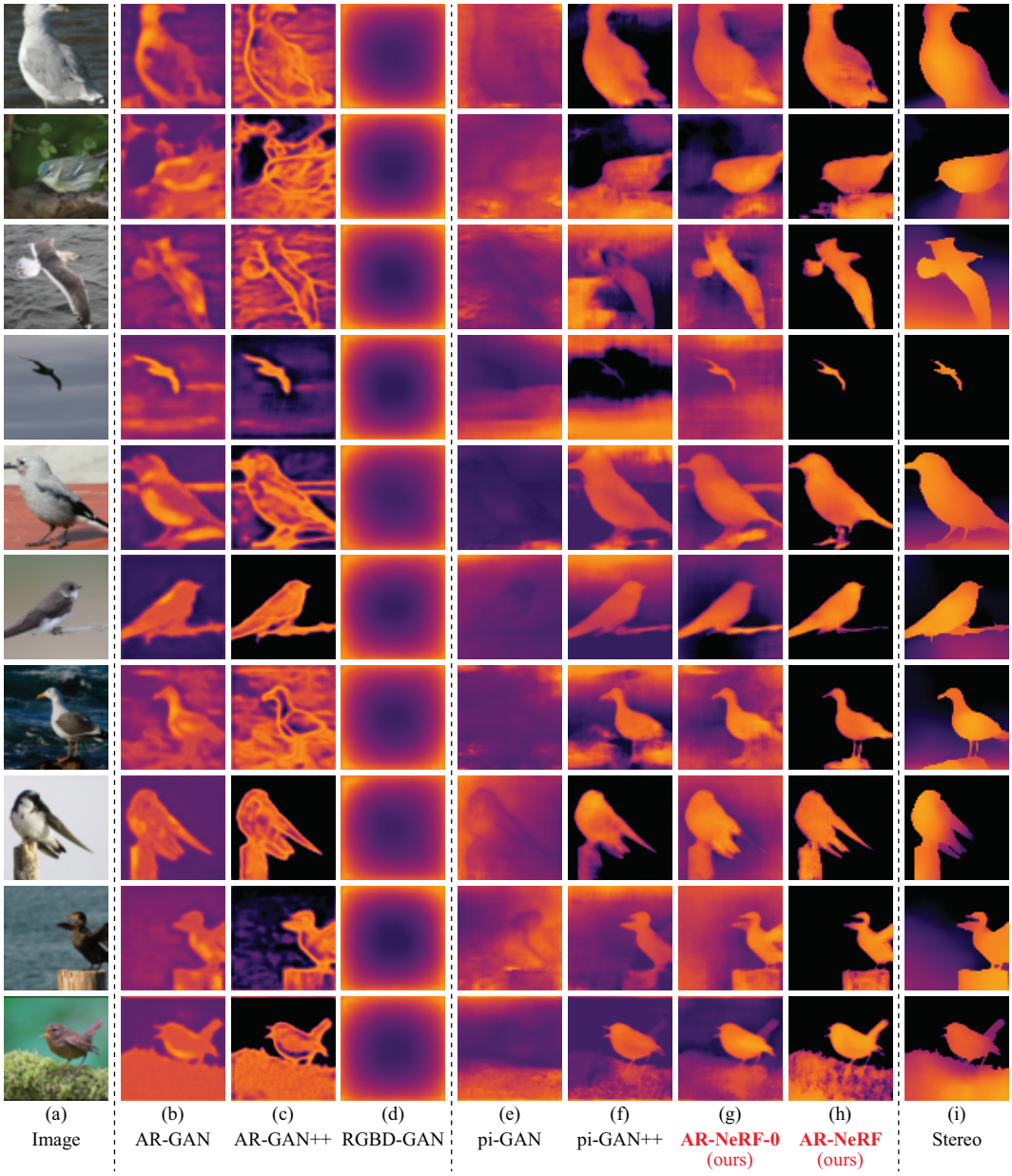


Figure 14. **Comparison of depth prediction on the CUB-200-2011 dataset.** These depths are used to calculate the SIDEs in Tables 1 and 2. AR-GAN (b), AR-GAN++ (c), and RGBD-GAN (d) are CNN-based and are trained in a *fully* unsupervised manner. In addition, pi-GAN (e), pi-GAN++ (f), AR-NeRF-0 (g), and AR-NeRF (h) are NeRF-based and trained in a *fully* unsupervised manner. By contrast, the model in (i) [26] was trained using *paired* supervision and was applied as the *ground truth* in the evaluation. The SIDEs in Tables 1 and 2 were calculated by comparing the depths in (b)–(h) with the depths in (i). Our findings are summarized as follows: (1) Because AR-GAN (b) and AR-GAN++ (c) only employ defocus (appearance) cues, their predicted depths are affected by their appearance. For example, in the fifth row, the horizontal boundary in the background is emphasized despite its non-necessity. (2) RGBD-GAN (d) utilizes viewpoint (geometric) cues for 3D representation learning. However, there are few viewpoint cues in this dataset; consequently, this model has difficulty in learning depth. (3) For the same reason, pi-GAN (e) and pi-GAN++ (f), which also only employ viewpoint cues, suffer from learning difficulty, although NeRF itself has a strong 3D consistency at the design level. In particular, they fail to consistently distinguish between the foreground (bird) and background (surroundings), parts of which are often mixed (e.g., in the third and fourth rows). (4) Although the results of AR-NeRF-0 (g) are closest to those of AR-NeRF (h), AR-NeRF-0 (g) is often affected by the appearance (e.g., in the fifth row, similar to AR-GAN (b) and AR-GAN++ (c)) because it also only leverages focus cues. (5) AR-NeRF (h) overcomes the limitations of pi-GAN (e), pi-GAN++ (f), and AR-NeRF-0 (g) by utilizing both the viewpoint and defocus cues.

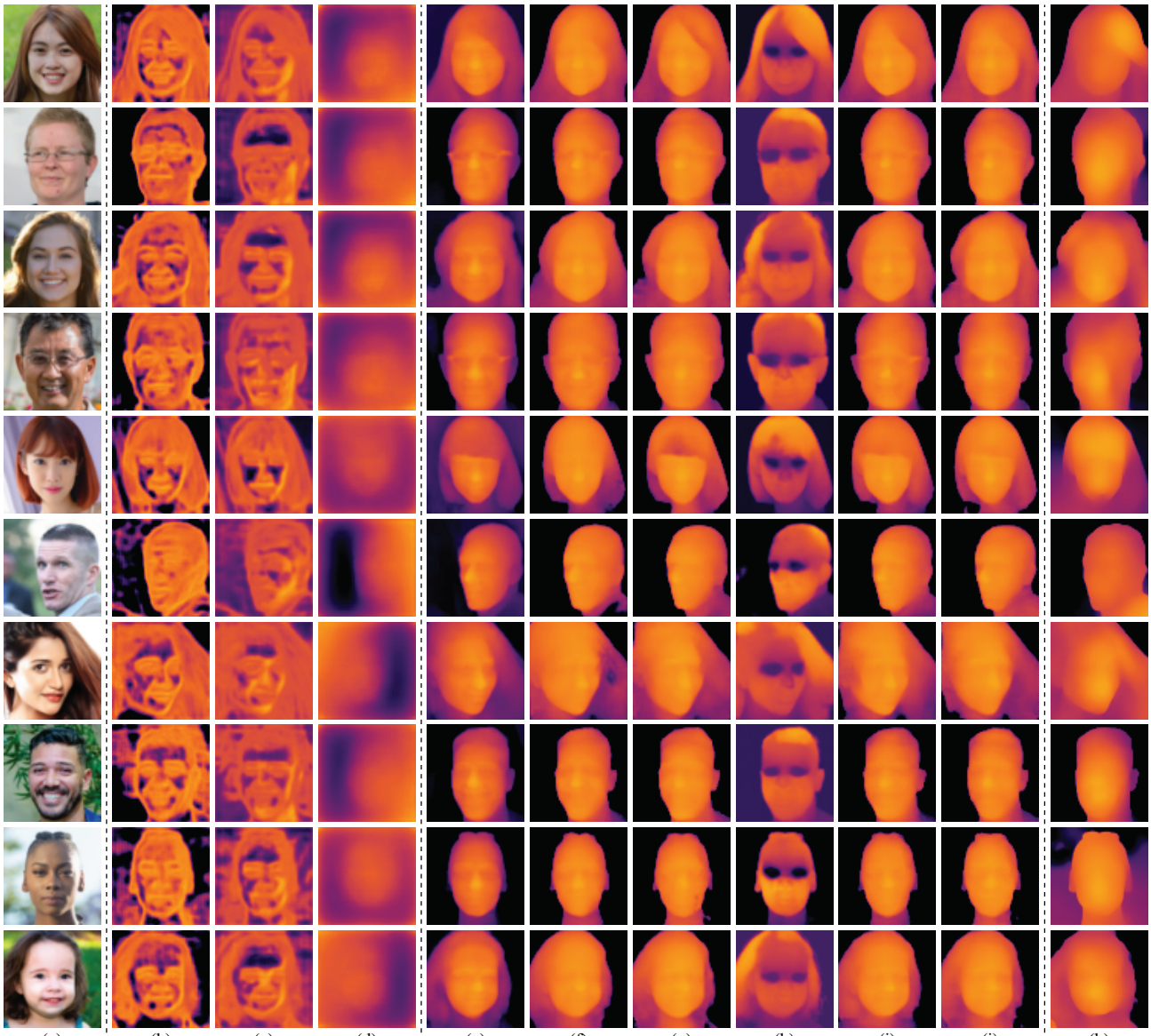


Figure 15. **Comparison of depth prediction on the FFHQ dataset.** These depths are used to calculate the SIDEs in Tables 1 and 2. AR-GAN (b), AR-GAN++ (c), and RGBD-GAN (d) are CNN-based and are trained in a *fully* unsupervised manner. In addition, pi-GAN (e), pi-GAN++ (f), pi-GAN++-F (g), AR-NeRF-0 (h), AR-NeRF (i), and AR-NeRF-F (j) are NeRF-based and trained in a *fully* unsupervised manner. By contrast, the model in (k) [26] was trained using *paired* supervision and was applied as the *ground truth* in the evaluation. The SIDEs in Tables 1 and 2 were calculated by comparing the depths in (b)–(j) with those in (k). Our findings are summarized as follows: (1) In the depths predicted using AR-GAN (b) and AR-GAN++ (c), holes appeared in the face regions. This is because they can only adopt a defocus cue, which is insufficient to distinguish a flat surface in a face from the blur caused by the defocus. (2) RGBD-GAN succeeded in capturing the face direction in depth by leveraging the viewpoint cue (e.g., in the sixth and seventh rows); however, the depth fidelity was low. This can occur because RGBD-GAN imposes 3D consistency only at a loss level and not at an architectural level. (3) By contrast, the NeRF-based models (e)–(j) have 3D consistency at the architectural level, allowing high fidelity and consistent 3D depths to be predicted. In particular, we found that the models utilizing viewpoint cues ((e)–(g), (i), and (j)) demonstrated similar performance. This is because this dataset includes sufficiently varying viewpoints. (4) However, AR-NeRF-0 (h), which can only use the defocus cue, fails to capture structures around the eyes. This is possibly because there is a large variety of appearances around the eyes, and it is difficult to model the corresponding depth using only a defocus cue. We can overcome this limitation by jointly using viewpoint and defocus cues, as in AR-NeRF (i) or AR-NeRF-F (j).

C. Implementation details

In this appendix, we provide implementation details regarding the following items:

- Appendix C.1: Details of the main experiments (Section 5).
- Appendix C.2: Details of defocus renderer (Appendix A.5).

C.1. Details of main experiments (Section 5)

C.1.1 Dataset

In the experiments, we used three datasets, the detailed information of which is as follows:

Oxford Flowers [19]. The dataset consists of 8,189 images with 102 flower categories. Each category includes 40 or more images. The images were obtained by searching the web and taking photographs. We downloaded the data from an official website.[3] More detailed information is provided in the README file available on the website.

CUB-200-2011 [24]. The dataset contains 11,788 images of 200 bird species. The images were collected using a Flickr image search and then filtered by presenting each image to multiple users of Mechanical Turk [25]. We downloaded the data from an official website.[4] More detailed information is provided in the technical report [24].

FFHQ (Flickr-Faces-HQ) [10]. The dataset consists of 70,000 face images. The images were crawled from Flickr. Therefore, as the dataset creators [10] mention, the dataset inherits all the biases of that website. The images were filtered using automatic filters and Amazon Mechanical Turk. Only images under permissive licenses (Creative Commons BY 2.0, Creative Commons BY-NC 2.0, Public Domain Mark 1.0, Public Domain CC0 1.0, or U.S. Government Works license) were collected. The dataset itself is available under a Creative Commons BY-NC-SA 4.0 license by the NVIDIA Corporation. We downloaded the data from an official website.[5] More detailed information is provided in the README file available on the website.

C.1.2 Network architectures

As explained in Section 5.1, we implemented AR-NeRF based on the pi-GAN [1],[6] which is a state-of-the-art generative variant of NeRF. Because the original pi-GAN was not applied to the datasets used in our experiments, we carefully tuned the configurations and hyperparameters so that the baseline pi-GAN could generate images sufficiently well. In particular, we used the configuration of CelebA [14][7] as the default and tuned depending on the dataset. We explain the details of each network below.

Mapping network. In pi-GAN, a StyleGAN [10]-inspired mapping network was introduced to efficiently propagate information in the latent code to each layer. We implemented this network using an MLP with three hidden layers (256 units each). We used leaky rectified linear units (LReLUs) [15] with a negative slope of 0.2 as activation functions. The dimension of the latent code was set to 256. This architecture is the same as that of the original pi-GAN [1].

Synthesis network. In pi-GAN, SIREN [23]-based implicit radiance fields are used as a synthesis network. We implemented this network using an MLP with eight FiLM [3,20]-SIREN hidden layers of 128 units each. In the original pi-GAN [1], 256 units were used in each layer; however, in our preliminary experiments, we found that the reduction in the units did not significantly affect the performance. In addition, this reduction allowed the use of a larger number of points along the ray, which is critical for improving performance. Considering this, we used 128 units in our experiments. In pi-GAN++ and AR-NeRF, we used the above-mentioned network as a foreground synthesis network and used a background synthesis network. We implemented the latter network using an MLP with eight FiLM-SIREN hidden layers of 64 units each. We used fewer parameters in the background synthesis network under the assumption that the background is simpler than the foreground.

Discriminator. We used different discriminators according to the dataset. For FFHQ, we used the same discriminator as that used in pi-GAN for CelebA.[7] The discriminator was implemented using CoordConv layers [13] and residual blocks [6]. In our preliminary experiments, we found that the CoordConv layers yield negative effects for Oxford Flowers and CUB-200-2011. A possible cause is that in FFHQ, faces are aligned based on facial landmarks, whereas in Oxford Flowers and CUB-200-2011, flowers and birds are not strictly aligned. Based on this finding, we removed the CoordConv layers from the discriminator when applied to Oxford Flowers and CUB-200-2011.

C.1.3 Training settings

We used different training settings according to the dataset. For FFHQ, we used the same setting as that in pi-GAN for CelebA.[7] More specifically, as the GAN objective, we used the non-saturating GAN loss [5] with real gradient penalty (R_1) regularization [16], where the weight parameter of the R_1 regularization was set to 0.2. Additionally, we used an

[3] https://www.robots.ox.ac.uk/~vgg/data/flowers/102/.
[4] http://www.vision.caltech.edu/visipedia/CUB-200-2011.html.
[5] https://github.com/NVlabs/ffhq-dataset.
[6] https://github.com/marcoamonteiro/pi-GAN.
[7] https://github.com/marcoamonteiro/pi-GAN/blob/master/curriculums.py.

identity regularizer [18] with the weight parameter of 15 to keep the identity across different viewpoints. The network was trained for 200,000 iterations using the Adam optimizer [11], with learning rates of 0.00006 and 0.0002 for the generator and discriminator, respectively, and momentum terms β_1 and β_2 of 0 and 0.9, respectively. The batch size was set to 16. We used an exponential moving average [9] with a decay of 0.999 over the weights to generate the final generator. In pi-GAN, we set the number of sample points along the ray to 48, where 32 and 16 points were used for stratified sampling and hierarchical sampling, respectively. In pi-GAN++ and AR-NeRF, we set the values for the foreground and background synthesis networks as 48 and 24, respectively. In the foreground synthesis, 32 and 16 points were used for stratified sampling and hierarchical sampling, respectively. In the background synthesis, 16 and 8 points were used for stratified sampling and hierarchical sampling, respectively. A field of view was set to 12°. As discussed in Section 5.3, we used the same number of rays in all models to investigate the pure performance differences between the models with and without aperture rendering. Specifically, we used five stratified sampled rays (Section 4.4) in AR-NeRF and five ensemble rays (i.e., five rays with an aperture size $s = 0$) in pi-GAN and pi-GAN++.

For Oxford Flowers and CUB-200-2011, we also used differentiable augmentation [28][8] to stabilize the training. In particular, we used color jittering, translation, and cutout [2] for Oxford Flowers, and translation for CUB-200-2011 because we found that they were the best choice. We removed the identity regularizer [18] because we found that it yields negative effects for Oxford Flowers and CUB-200-2011. The other settings were the same as those for FFHQ.

C.1.4 Evaluation

As described in Section 5.1, we calculated KID using 20,000 generated images and all real images. We implemented a KID calculator based on the official code.[9] We implemented the depth predictor, which was trained with images and depth generated by GANs, and used it to calculate the SIDE using U-Net [21]. In particular, we used the same network and training settings as those used in the AR-GAN study [8] for direct comparison. pi-GAN++ and AR-NeRF can synthesize the unbounded background (or depth) by using a NeRF++ [27]-based background synthesis network; however, the predictable depth range is bounded in a typical depth predictor, including the model [26] that was used as "ground truth" in our experiment. Therefore, we used only the foreground synthesis network when calculating the SIDE.

[8] https://github.com/mit-han-lab/data-efficient-gans.

[9] https://github.com/mbinkowski/MMD-GAN.

C.2. Details of defocus renderer (Appendix A.5)

C.2.1 Dataset

In the experiment, we used a test set of the *iPhone2DSLR Flower* [29] for the evaluation. Its detailed information is as follows:

iPhone2DSLR Flower [29]. The dataset includes 2,381 smartphone images and 3,805 DSLR images. The smartphone images were collected from Flickr by searching for photos taken by Apple iPhone 5, 5s, or 6, with the search text "flower." DSLR images with a shallow depth-of-field (DoF) were also collected from Flickr using the search tags "flower" and "dof." We downloaded the data from an official website.[10] More detailed information is provided on the website and appendix of the corresponding paper [29].

C.2.2 Network architectures

We implemented AR-NeRF-R using basically the same network as AR-GAN-DR [8], that is, we used the U-Net architecture [21]. A difference from AR-GAN-DR is that we extended U-Net to a conditional setting [30]. Specifically, we injected the aperture size s and focus distance f into every intermediate layer in the encoder after expanding them to the corresponding feature map size.

C.2.3 Training settings

We generated training data (i.e., pairs of all-in-focus and focused images with auxiliary information on aperture size s and focus distance f) using AR-NeRF, which was trained using 64×64 images on the Oxford Flowers dataset. AR-NeRF was the same as that used to generate the samples in Figures 1 and 3. When training AR-NeRF-R, we used 128×128 images generated by AR-NeRF, where we increased the resolution of the generated images from 64×64 to 128×128 by increasing the density of input points (Section 5.2). We trained the defocus renderer for 300,000 iterations using the Adam optimizer [11] with a learning rate of 0.0003 and momentum terms β_1 and β_2 of 0.9 and 0.99, respectively. The batch size was set to 4. The learning rate was kept constant during training, except for the last 30% iterations, where the learning rate was smoothly ramped down to zero.

[10] https://github.com/junyanz/CycleGAN.

References

[1] Eric R. Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *CVPR*, 2021. 2, 16

[2] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 17

[3] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 3(7):e11, 2018. 16

[4] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 3

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 16

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 16

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a Nash equilibrium. In *NIPS*, 2017. 3

[8] Takuhiro Kaneko. Unsupervised learning of depth and depth-of-field effect from natural images with aperture rendering generative adversarial networks. In *CVPR*, 2021. 2, 3, 17

[9] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2017. 17

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 16

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 17

[12] Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in GANs. In *ICML*, 2019. 3

[13] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In *NeurIPS*, 2018. 16

[14] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 16

[15] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshops*, 2013. 16

[16] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *ICML*, 2018. 16

[17] 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. In *ECCV*, 2020. 2

[18] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. In *ICCV*, 2019. 17

[19] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008. 16

[20] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 16

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2, 17

[22] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. In *NeurIPS*, 2020. 2

[23] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. 16

[24] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 16

[25] Peter Welinder, Steve Branson, Pietro Perona, and Serge Belongie. The multidimensional wisdom of crowds. In *NIPS*, 2010. 16

[26] Ke Xian, Jianming Zhang, Oliver Wang, Long Mai, Zhe Lin, and Zhiguo Cao. Structure-guided ranking loss for single image depth prediction. In *CVPR*, 2020. 13, 14, 15, 17

[27] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 17

[28] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In *NeurIPS*, 2020. 17

[29] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 3, 17

[30] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017. 2, 17