Dynamic NeRF on RGB-D Data

Johannes Kirmayr
johannes.kirmayr@tum.de

Philipp Wulff
philipp.wulff@tum.de

Abstract

Current techniques for novel view synthesis using neural radiance fields (NeRF) in non-rigid 3D scenes produce a scene representation that matches the ground truth with a noticeable error. Our approach reduces this error by improving and extending existing dynamic NeRF by depth using an RGB-D dataset. We constrain the optimization by depth supervision and depth-guided sampling (DGS). Our method outperforms current techniques in rendering novel views of the learned scene representation. We provide the code, data and videos of our results on our website.

1. Introduction

Rendering scenes from a free point of view is used in many current virtual/augmented reality applications. Conventional methods use expensive multi-view capturing of the scene. Latest approaches that use Neural Rendering [11] show photorealistic results while saving memory space in comparison to often used voxel grids. While the original NeRF [11] is made for static scenes, neurally rendering a realistic, dynamic scene from a given input video, is more challenging. As a solution, recent approaches [12, 13, 17] disentangle the scene into a rigid canonical scene and its non-rigid deformation. However, it remains difficult to resolve the ambiguity of dynamic scenes, resulting in non-photo-realistic renderings. We leverage depth from commonly available RGB-D sensors to guide NeRF optimization and improve color and geometry for novel view synthesis on deforming scenes. In summary, our contributions are:

- Supervising D-NeRF [13] by a depth loss and extending it by depth-guided sampling
- Benchmarking latent deformation codes and the rigidity network from [12, 17];
- Creation of a real-world RGB-D dataset

2. Related Work

Immersive virtual experiences in real-world scenes require the rendering of novel views. If densely captured observation images of a scene are available, light field interpolation [5, 7] can synthesize novel views. With fewer observations, representations which use information about the scene geometry perform better, such as triangle-meshes [2, 3, 6], voxel grids [9], or multiplane images [4, 10]. Recently, NeRF [11] introduced a well-performing approach that represents a scene as a continuous volumetric model, using a MLP to minimize the loss of rerendering all observed views. Many recent methods extend NeRF's volumetric representation for novel view synthesis.

D-NeRF [13], NR-NeRF [17] and Nerfies [12] extend NeRF's volumetric model to dynamic scenes. Herein, they process dynamic scenes in two stages: first, a deforming or ray-bending network maps points in the deformed scene to corresponding points in a shared canonical scene; followed by the original static NeRF architecture. Optimizing a dynamic scene representation for novel view synthesis from monocular input views is a severely underconstrained problem [12, 17]. For this reason, additional constraints are introduced. In NR-NeRF the ray bending offsets are multiplied with a rigidity-mask as a means of disabling the deformation of static elements in the scene. Both, NR-NeRF and Nerfies add deformation-based regularizers besides the color reconstruction loss and latent deformation codes.

Rendering novel views with NeRF requires an accurate representation of the scene geometry in the volumetric model. Recent work shows that conditioning NeRF on dense depth priors [14] improves the novel view renderings and allows for using fewer observations. In [14] the authors constrain NeRF by constructing a depth loss and the use of depth guided sampling.

3. Method

We introduce our method: Depth-Guided-Dynamic NeRF (DGD-NeRF). It extends the D-NeRF framework [13] by including depth maps besides the corresponding RGB images (Sec. 3.1 and Sec. 3.3). Moreover, we implement latent deformation codes and rigidity network from related works (Sec. 3.1). Each extension in our modular pipeline can be disabled - Fig. 1 shows the complete optimization architecture.

We use a RGB-D dataset with RGB images $\{\boldsymbol{C}_i, t_i\}_{i=0}^{N-1}$, where $\boldsymbol{C}_i \in [0,1]^{H\times W\times 3}$ denotes color val-

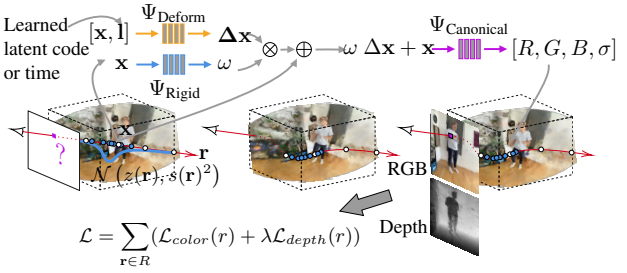

Figure 1. **DGD-NeRF pipeline.** The model consists of three MLPs: a deformation network mapping deformations to a shared canonical volume; a rigidity network masking deformations with rigidity scores; and a canonical volume regressing radiance and volume density from every camera ray.

ues of an image taken at time $t_i \in [0, 1]$ under camera pose $P_i \in \mathbb{R}^{4\times4}$, and a depth map $\boldsymbol{D}_i \in [h_t, h_f]^{H\times W}$ where h_n and h_f are the near and far planes of the canonical volume.

3.1. Network Architecture

The architecture shown in Fig.1 consists of three MLPs: the *deformation network* Ψ_{Deform}, the *canonical network* $\Psi_{\text{Canonical}}$ and the *rigidity network* Ψ_{Rigid}, which is optional.

Canoncial Network: It is computationally unfeasible to train a separate NeRF model for each point in time of a dynamic scene. Therefore, we consider the state of the 3D Volume at some time $t > 0$ as a deformation from one *static* canonical scene, which is shared for all points in time. In practice, the canonical configuration is set as the scene state at $t = 0$. This allows training a single canonical network $\Psi_{\text{C}}(\boldsymbol{x}) \mapsto (\boldsymbol{c}, \sigma)$ to map a 3D point $\boldsymbol{x}$ in the canonical volume to its color $\boldsymbol{c}$ and volumetric density σ. Before passing $\boldsymbol{x}$ to Ψ_{C}, we apply NeRF's positional encoding [11, 16].

Deformation Network: $\Psi_D(\boldsymbol{x}, t) \mapsto \triangle\boldsymbol{x}$ predicts the deformation $\triangle\boldsymbol{x}$ of a 3D point from the deformed scene at time t to the canonical configuration at time $t = 0$. Herein, the inputs to the network, the location $\boldsymbol{x}$ and time t, are again positionally encoded as in [13].

Extension: Learnable Latent Codes Instead of feeding the positionally encoded time t to the deformation network, we add the option to supply a per-image, learnable, latent deformation codes as it is done in related works [14, 17]. These are passed to the deformation network and optimized jointly with the network weights via gradient descent. At test time, novel points in time are achieved by interpolation between latent codes.

Extension: Rigidity Network As introduced in [17] we implemented a self-learning 3-layer MLP Ψ_{Rigid} which assigns a rigidity score $\omega \in [0, 1]$ to each point $\boldsymbol{x}$ independent of the time t. This score masks the predicted deformation $\triangle\boldsymbol{x}$, effectively disabling any deformation for $\omega = 0$. The intuition is that the rigidity network shares information across all points in time about which volumes are rigid such that the deformation network can focus on non-rigid volumes.

3.2. Volumetric Rendering

In combination, the MLPs map a 3D point at time t to its color $\boldsymbol{c}$ and density σ. To get novel view images, we generate a 2D projection of the deformed scene as seen from the camera pose $P_i \in \mathbb{R}^{4\times4}$ by rendering the color of each pixel individually, as shown in the left box in Fig. 1. In order to render a pixel p we shoot a ray $\boldsymbol{r} = \boldsymbol{r}_{\text{o}} + h \cdot \boldsymbol{r}_d,\ h \in [0, h_f]$, through the 3D Volume, where $\boldsymbol{r}_{\text{o}}$ is the ray's origin and $\boldsymbol{r}_{\text{d}}$ is the ray direction. Next, we distribute $\{h_i\}_{i=1}^N$ sampling locations along the ray between the near plane $h_n = 0.1$ and far plane $h_f = \text{max\_depth} + 0.1$. The deformation to the canonical volume effectively bends the ray, as shown in the central box in Figure 1. The color $\hat{C}$ of pixel p at time t is then approximated with numerical quadrature of the volume rendering integral along the ray $\boldsymbol{r}$:

$$\begin{aligned} \hat{\boldsymbol{C}}(\boldsymbol{r}) &= \sum_{i=1}^{N} \mathcal{T}_i' \alpha_i \mathbf{c}_i, \\ \text{where} \quad \mathcal{T}_i' &= \exp\left(-\sum_{m=1}^{i-1} \sigma_m \delta_m\right) \\ \alpha_i &= 1 - \exp(-\sigma_i \delta_i) \\ \boldsymbol{p}_i &= \mathbf{x}(h_i) + \Psi_{\text{Deform}}(\mathbf{x}(h_i), t) \\ [\mathbf{c}_i, \sigma_i] &= \Psi_{\text{Canonical}}(\mathbf{p}_i), \end{aligned} \tag{1}$$

and $\delta_i = h_{i+1} - h_i$; the color predictions of the samples are weighted by transmissibility and density.

Extension: Depth-Guided Sampling (DGS) D-NeRF [13] uses hierarchical sampling, that is it first determines a density distribution along the ray using a forward pass with "coarse" samples placed in stratified locations along the ray and then uses this distribution to draw additional "fine" samples in locations of high density. DGS aims to improve the selection of fine samples by directly drawing the fine samples from a Gaussian $\mathcal{N}(z(\boldsymbol{r}), s(\boldsymbol{r}))$ centered on the ray's depth $z(\boldsymbol{r})$ with some variance $s(\boldsymbol{r})$. At train time, we use the measured depth and sensor uncertainty, whereas at test time a depth estimate $\hat{z}(\boldsymbol{r})$ and its uncertainty $\hat{s}(\boldsymbol{r})$ is obtained from a forward pass with the first set of coarse samples. The depth estimate and standard deviation can be computed from the rendering weights α_i [14]:

$$\hat{z}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i h_i, \quad \hat{s}(\mathbf{r})^2 = \sum_{i=1}^{N} \alpha_i \left(h_i - \hat{z}(\mathbf{r})\right)^2. \tag{2}$$

DGS places samples around the area of interest - the object's surface.

3.3. Network Training

All parameters of the three MLPs (ψ_C, ψ_D, ψ_R) are optimized jointly. A mean squared error (MSE) **color loss** is applied per ray $\boldsymbol{r}$: $\mathcal{L}_{\text{color}}(\boldsymbol{r}) = ||\hat{\boldsymbol{C}}(\boldsymbol{r}) - \boldsymbol{C}(\boldsymbol{r}))||_2^2$, where $\boldsymbol{C}$ is the ground truth color.

Extension: Depth Loss We add a depth loss to the color loss as an additional constraint on the optimization problem. Herein, we try two types - a MSE depth Loss:

$$\mathcal{L}_{depth}(\boldsymbol{r}) = ||\hat{z}(\boldsymbol{r}) - z(\boldsymbol{r})||_2^2 \tag{3}$$

and a Gaussian negative log likelihood (GNLL) depth loss [14]:

$$\mathcal{L}_{depth}(\boldsymbol{r}) = \begin{cases} \log((\boldsymbol{r})^2) + \frac{(\hat{z}(\boldsymbol{r}) - z(\boldsymbol{r}))^2}{\hat{s}(\boldsymbol{r})^2} & \text{, if P or Q} \\ 0 & \text{, otherwise} \end{cases} \tag{4}$$

which is non-zero when when $P = |\hat{z}(\boldsymbol{r}) - z(\boldsymbol{r})| > s(\boldsymbol{r})$ or $Q = \hat{s}(\boldsymbol{r}) > s(\boldsymbol{r})$. In theory, the GNLL depth loss supports depth-guided sampling as it drives the minimization of the uncertainty of the depth estimate.

The complete Loss term becomes:

$$\mathcal{L} = \sum_{\boldsymbol{r}} \left(\mathcal{L}_{\text{color}}(\boldsymbol{r}) + \lambda \mathcal{L}_{depth}(\boldsymbol{r}) \right). \tag{5}$$

4. Results

We evaluate our model with experiments (Sec. 4.2) on a custom dataset (Sec. 4.1) and compare it against D-NeRF [13] as a baseline.

4.1. Experimental Setup

Dataset Initially, we used the DeepDeform dataset [1] with RGB-D videos of deforming scenes captured with a *static* camera. However, the presence of many invalid depth pixels is problematic for DGD-NeRF and the static camera pose does not permit quantitative evaluations of novel view renderings. Therefore, we built a custom RGB-D dataset of challenging scenes. We used an iPad to record RGB-D videos and the corresponding camera poses. Each scene contains between 92 to 102 frames which are split into the different sets (70-15-15). The videos are captured with 15 or 6 fps respectively.

NeRF Optimization We train on ray batches of 512 or 1024 and use the Adam optimizer with default parameters as well as a learning rate of 0.0005 with exponential decay (decay rate: 0.1 with 250k step size). Each model in the ablation experiments is trained for 400k iterations, taking between 10 and 20 hours on a Nvidia GTX 1080Ti dependent on the experiment. Regardless of the sampling strategy, rays sampled in 192 or 96 locations with $1/3$ coarse samples and $2/3$ fine samples. The MSE depth loss is weighted by 0.1 and the GNLL depth loss by 0.001.

	Method	PSNR ↑	SSIM ↑	LPIPS ↓	Depth-RMSE ↓
Human	D-NeRF [13]	21.31	0.753	0.382	0.533
	w/ MSE	21.63	0.766	0.361	**0.112**
	w/ GNLL	21.35	0.745	0.390	0.208
	w/ DGS	21.45	0.764	0.374	0.566
	w/ LDC	16.53	0.591	0.560	0.921
	w/ RN	21.16	0.748	0.386	0.568
	w/ MSE + DGS	21.49	**0.767**	**0.354**	**0.112**
	w/ GNLL + DGS	**21.71**	0.762	0.378	0.232
Bottle	D-NeRF [13]	15.61	0.575	0.570	0.309
	D-NeRF s/2	15.47	0.580	0.567	0.297
	w/ MSE + DGS	16.12	0.597	0.580	0.090
	w/ MSE + DGS s/2	**16.34**	**0.612**	**0.578**	**0.086**
Gobblet	D-NeRF [13] s/2	20.61	0.700	0.537	0.333
	w/ MSE + DGS s/2	**21.20**	**0.718**	**0.521**	**0.028**

Table 1. **Quantitative comparison** of different experiments on three scenes from our own dataset.

Evaluation Metrics Besides qualitative observations, we compute the peak signal-to-noise ratio (PSNR), the Structural Similarity Index Measure (SSIM) [18] and the Learned Perceptual Image Patch Similarity (LPIPS) [19] to measure the quality of the RGB novel view renderings. We assess the accuracy of the expected ray-termination depth of our method using the root-mean-square error (RMSE) against the sensor depth – measured in meters.

4.2. Experiments

We evaluate our model on test frames from three scenes and benchmark the quantitative performance gain of each extension compared to the baseline with the *Human* scene. Based on this, we show quantitative (Tab. 1) and qualitative results (Fig. 2) of our best model on all three scenes. In the *Bottle* scene we down-sample the training frames to 6 fps and test halving the number of samples along the rays. In the *Gobblet* scene we increase the fps again.

Human Scene The scene has 15 fps, we use 500 rays per batch and 192 samples per ray.

With Depth Loss: The models on the Human scene

with an MSE resp. GNLL depth loss improve most image metrics over the baseline, particularly the depth estimate (Tab. 1). MSE returns better results than GNLL.

With DGS: Standalone DGS improves the image metrics slightly. However, as depth is not supervised, the depth RMSE stays large.

With Learnable Deformation Codes or Rigidity Network: Both extensions from [17] show no improvement in the quantitative results for our experiment (Tab. 1).

With Depth Loss and DGS: Combining both extensions improves all image metrics further. Using the MSE loss is preferable over GNLL, even though DGS results in a relatively larger improvement to the model that only uses a GNLL depth loss compared to MSE (Tab. 1). We hypothesize, this is because GNLL minimizes the standard deviation of the depth prediction.

Bottle Scene The scene has 6 fps, we use 1024 rays per batch and 192 or 96 samples per ray. The input video is "shaky", as we wanted many angles of the scene. However, we believe that the the iPad retrieves imprecise pose measurements which negatively impacts our overall results.

With Fewer Fps: Reducing from 15 to 6 fps leads to a significant decrease in performance of both models (Tab. 1). Our model still outperforms the baseline, but struggles with the larger spatio-temporal changes in the input frames.

With Half Samples: Halving the samples along a ray does not change the results for either model (Tab. 1). Therefore, we go on in the next experiment *Gobblet* with half the samples as default.

Gobblet Scene The scene has 15 fps, we use 1024 rays per batch and 96 samples per ray.

With Half Samples: Our model with an MSE depth loss and DGS outperforms the baseline, supporting our results on the *Human* scene (Tab. 1).

Qualitative Results Our model used for the qualitative results in Fig. 2 uses the MSE depth loss and depth-guided sampling as this performed the best. We use 192 samples in the *Human* scene, 96 samples in the *Bottle* scene, and 48 samples in the *Gobblet* scene. The images are consistent with the quantitative results, for example in row 1 it can be seen that our depth prediction is much more accurate due to the depth loss. This leads to fewer artefacts in the test set images (hence examples 1 and 2, Fig. 2). Furthermore, the addition of the depth loss to the color loss has no negative influence on the color reconstruction and produces comparable RGB predictions of both models. The data-efficient depth-guided sampling strategy particularly improves the renderings with fewer samples and larger spatio-temporal changes as in the novel view rendering in the *Gobblet* scene (example 3, Fig. 2). Here, the depth images show the largest difference in rendering quality between our model and the baseline D-NeRF [13].

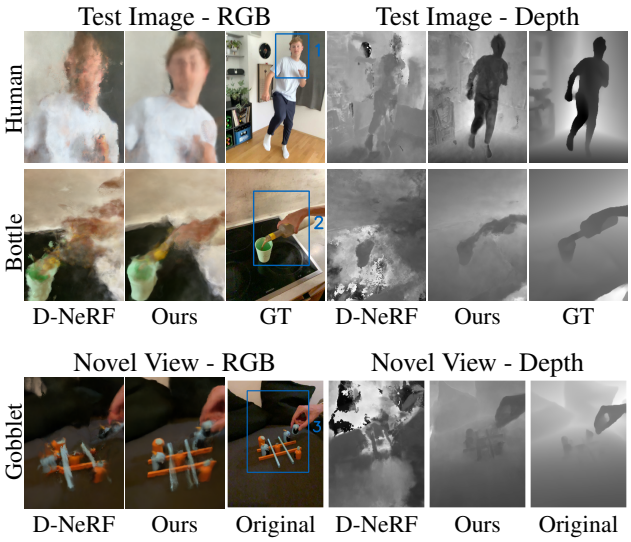

Figure 2. **Qualitative comparison** of test images and novel view renderings with D-NeRF [13] against our best model and the ground truth resp. the original image at the corresponding time.

5. Limitations

Despite improvements over the baseline, spatio-temporal changes in the scene are problematic. Thus, DGD-NeRF only provides photo-realistic results for small deformations. We are also limited by NeRF's long training and rendering times. These are reduced by usage of fewer samples, but remain several hours long for seconds-long scenes.

5.1. Future Work

The camera poses which we get from the iPad should be improved. Related works use the *structure from motion* method COLMAP [15] to do this. Moreover, we can estimate the rigidity of points in space when multiple rays at different times t hit particles at the same 3D locations. This information can then be fed as an additional input to the rigidity network. Finally, *Neural Sparse Voxel Fields* [8] promise improvements in training time of dynamic NeRF.

6. Conclusion

We have presented DGD-NeRF, a method to render novel spatio-temporal views using NeRF on a RGB-D dataset. While using a custom dataset, we demonstrated improvements over our baseline D-NeRF [13] through the use of depth data to constrain and guide the optimization. Considering the ease of retrieving RGB-D videos with modern handhelds, we believe that including depth into NeRF is an important step for improving its application.

References

[1] Aljaž Božič, Michael Zollhoefer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. 12 2019. 3

[2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001. 1

[3] Paul E. Debevec, Camillo Jose Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 1

[4] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 1

[5] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 43–54, New York, NY, USA, 1996. Association for Computing Machinery. 1

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

[7] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. 1

[8] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. 7 2020. 4

[9] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 1

[10] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 1

[11] 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. 1, 2

[12] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. 11 2020. 1

[13] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 3, 4

[14] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 1, 2, 3

[15] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 4

[16] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 2

[17] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhoefer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021. 1, 2, 4

[18] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 3

[19] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3