Supplementary Material: Consistent Multimodal Generation via A Unified GAN Framework

Zhen Zhu
UIUC
zhenzhu4@illinois.edu

Yijun Li
Adobe Inc.
yijli@adobe.com

Weijie Lyu
UIUC
wlyu3@illinois.edu

Krishna Kumar Singh
Adobe Inc.
krishsin@adobe.com

Zhixin Shu
Adobe Inc.
zshu@adobe.com

Sören Pirk
Adobe Inc.
soeren.pirk@gmail.com

Derek Hoiem
UIUC
dhoiem@illinois.edu

A. More Details About Method

While StyleGAN3 [6] uses critical sampling in the last two layers of the generator to ensure a good balance between antialiasing and the retention of high-frequency details, we maintain the same design in the RGB branch whereas we turn off this option for depth and normal branches since high-frequency details are not desired for these two modalities. StyleGAN3 provides two configurations: StyleGAN3-T and StyleGAN3-R. Compared to StyleGAN3-T, StyleGAN3-R replaces 3×3 convolutions with 1×1 convolutions and doubles the channel dimension to compensate for the lost capacity. Besides, it uses a radially symmetric jinc-based downsampling filter to replace the sin-based one. Though in practice, we noticed that StyleGAN3-R introduces many kaleidoscope-like patterns in results for the use of symmetric filters, this configuration achieves better performance on Stanford2D3D than StyleGAN3-T in terms of FID. Therefore, we use StyleGAN3-R by default. We follow StyleGAN3 [6] to disable style mixing and path length regularization as they introduce extra difficulties in convergence for complex datasets. We also blur all modalities to discriminators using a Gaussian filter with $\sigma = 10$ pixels over the first 200k images. As noted in [6], this prevents the discriminator from focusing too heavily on high frequencies as training starts.

For data augmentation operations, we follow ADA [5] to use pixel blitting (x-flip, 90° rotations, integer translation), general geometric translations (isotropic scaling, arbitrary rotation, anisotropic scaling, fractional translation), color transformations (brightness, contrast, luma flip, hue rotation, saturation), image-space filtering and image-space corruptions (additive noise, cutout). Since only color transformations require different processing for different channels while other operations can work on individual channels, we only apply color transformations for the RGB modality.

B. Implementation Details

The original Stanford2D3D dataset [1] provides training and validation splits. Training an unconditional generator does not explicitly require the split. Therefore, we use all data tuples from the dataset to train our Full models. The image resolution is up to 1024×1024. For quicker experiments, we resize all modalities to 256×256. For RGB, it would be transformed to the range of [-1, 1]. The Stanford2D3D dataset stores depth within the range between 0 and 65,535. To cope with such large depth range, we perform max rescaling by:

$$\mathrm{d}' = \left(\frac{\mathrm{d} - \mathrm{d.min}()}{\mathrm{d.max}()} - 0.5 \right) * 2, \tag{1}$$

so that the ground-truth for the depth modality lies within [-1, 1]. The original data of depth contains large areas of blank holes, so we use the hole filling method [10] to inpaint the holes for better generation quality. The original Stanford2D3D dataset saves surface normal as RGB images and we maintain the same procedure as RGB images when dealing with normals. So the channels for RGB, depth and surface normal are 3, 1, and 3, respectively.

We use a batch size of 4 for each GPU and 8 A100 GPUs using PyTorch 1.10.0 of CUDA 11.3 for most experiments. We use Adam optimizer [8] to optimize both generator and discriminators with $\beta 1 = 0$, $\beta 2 = 0.99$ and $\epsilon = 10^{-8}$. Following StyleGAN3 [6], we also use equalized learning rate for all trainable parameters [4], minibatch standard deviation layer at the end of the discriminator [4], exponential moving average of generator weights [4], mixed-precision FP16/FP32 training [5] to facilitate training, R1 regularization [9], and lazy regularization [7] to stabilize training.

C. More details about depth estimation and surface normal estimation

We build RGB→DEPTH and RGB→DEPTH* comparison methods, use LeReS [11] to evaluate SIE and perform depth estimation using the combination of real and generated data. When building RGB→DEPTH, we pretrain LeReS [11] on the training set created in Sec. 5.3 while using the validation set to choose the best performing model as the final supervision model. While evaluating the SIE, we use its pretrained ResNeXt101 model[1]. We follow the provided training script[2] to train the model using the same configurations on 8 A100 GPUs for 30K iterations. We use the same procedure to train depth estimation models using real and generated data unless different number of training data. For RGB→DEPTH*, we use the same pretrained ResNeXt101 model as in evaluating SIE. When evaluating the metrics of surface normal estimation, we use pre-trained models downloaded using the official script [2, 3][3].

D. More results

D.1. More Qualitative Results

In Sec. 4.3 of the manuscript, we show that our approach of training multiple modalities together gives better performance than RGB→DEPTH and RGB→DEPTH* which are trained on the RGB modality first and then move to depth using direct supervised learning. Aside from the difficulty of retrieving information from the features learned from RGB images for depth synthesis, an obvious drawback of this approach is the dependence on pre-trained models to provide supervisions. If the pre-trained models behave poorly on the target domain, learning from those models is not reliable. We show in Fig. 1 that when compared to pre-trained models, our generated depth (2nd column) and surface normal prediction (5th column) are better in capturing correct depth ranges and presenting clear layouts. The pre-trained depth estimator (3rd column) gives incorrect depth ranges and the surface normal estimator (final column) suffers from local blurriness. Even after we train the depth estimator on the whole Stanford2D3D dataset and then apply the depth estimator to provide supervision, the results (4th column) are still inferior to ours. In a sense, it is reasonable, since our approach directly learns from real ground-truth annotations while using pre-trained models (e.g., depth estimator) assumes learning from synthetic predicted annotations which could be unreliable.

We have two accompanying videos entitled "**RGB+depth.mp4**" and "**RGB+depth+normal.mp4**" under the "**videos**" folder to show the generated results of our model trained on {RGB, depth} and {RGB, depth, normal}, respectively. The results show the smoothness of transitions across different scenes and the consistency of different modalities. Even without clear guidance of camera poses, our model is still able to interpolate reasonably in a particular scene.

For each video clip, we sample 11 distinct latent codes, z, projecting them to corresponding intermediate latent codes, w. We perform a linear interpolation between two successive ws at a frame rate of 60, generating the associated RGB images and other modalities with the generator. The video is composed by compiling these sequential frames.

D.2. Consistent RGBD Generation Video

We have provided a video titled "**consistent-rgbd-generation.mp4**" in the "**videos**" directory to demonstrate the consistency between our generated depths and RGB images. Conversely, the depth estimations based on RGB images using the off-the-shelf depth estimator [11] present inconsistencies, either locally or globally, across frames.

D.3. Cross-domain Fine-tuning Video

As described in Sec. 5.2 of the manuscript, we can hold out a particular scene from the Stanford2D3D dataset, i.e., *auditorium*, to pre-train our model and then fine-tune our model on the held-out scene by using only a few pairs and the rest RGB images. During fine-tuning, we observe that data augmentation for discriminators would cause incorrect layouts in the results, e.g., chairs in the same auditorium scene facing towards opposite directions. We assume when the number of pairs drastically decreases, it also decreases the chances for the discriminator to see real data and tuples. Hence, the distorted data is leaked to the generator, causing incorrect geometries. We hence remove the ADA for fine-tuning. These models are fine-tuned using around 10K training iterations, which is around 1/78 of pre-training iterations.

We provide one accompanying video entitled "**finetune.mp4**" under the "**videos**" folder, which shows the result of the pre-trained model and then the models trained on different portions of pairwise data. Note that we use the same latent code z to generate those videos and observe that though these are different models, the results at the same time step interestingly have related structures, indicating that after fine-tuning, some initial properties still remain. The transition is quite smooth and the quality is decent for the fine-tuning results, considering that we only use a small number of pairwise data.

[1] https://cloudstor.aarnet.edu.au/plus/s/lTIJF4vrvHCAI31

[2] https://github.com/aim-uofa/AdelaiDepth/tree/main/LeReS

[3] https://github.com/EPFL-VILAB/omnidata/blob/2a661c93285018b71141759d2a1ad53d8aed0e62/omnidata_tools/torch/tools/download_surface_normal_models.sh

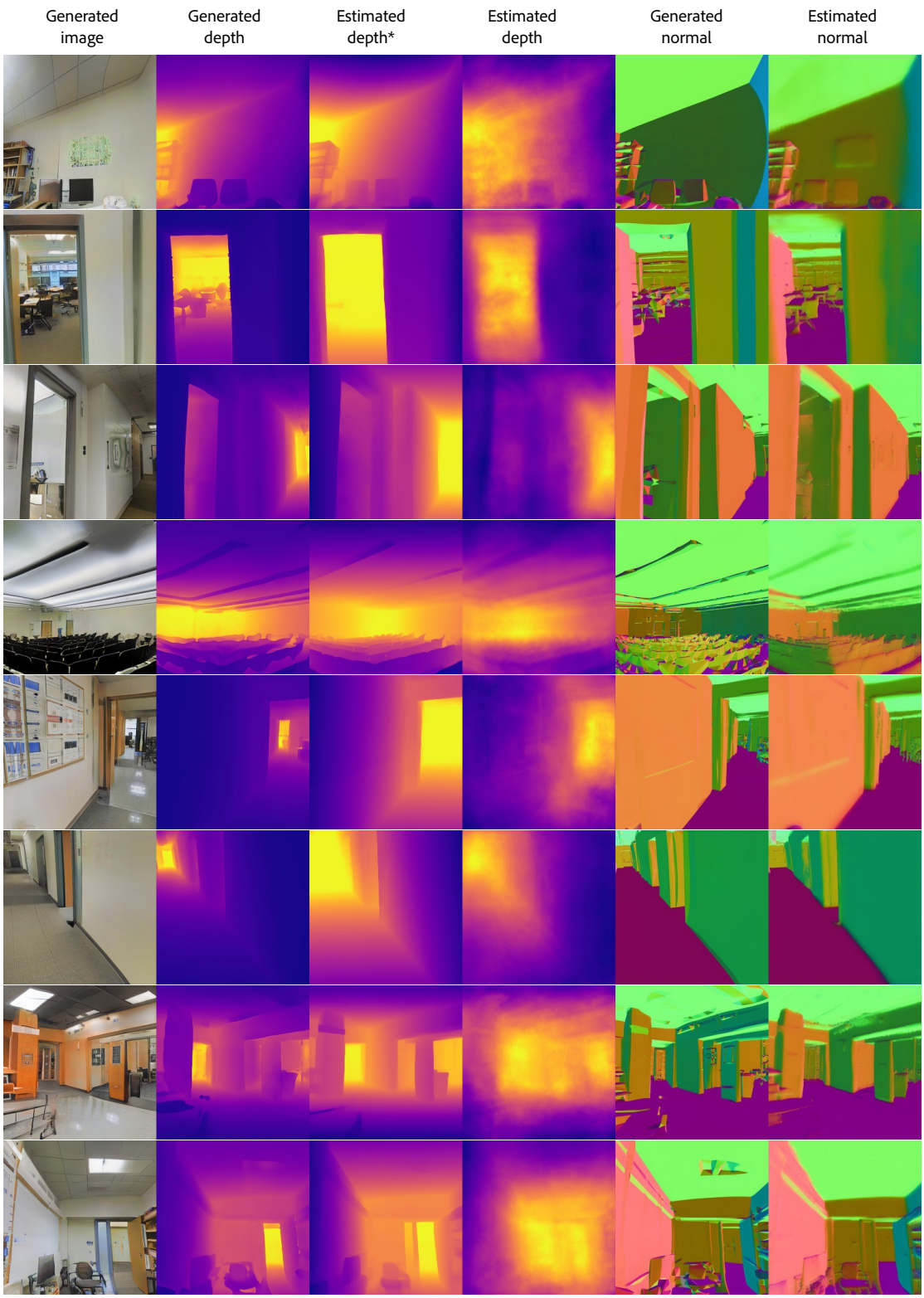

Figure 1. Demonstration of the comparison of our generated depth and surface normal with pre-trained depth estimators and normal estimator. The first column, second column and 5th column are generated RGB images, depths and normals from our model. The third column (Estimated depth*) represents the results estimated by pre-trained LeReS [11] by feeding the generated RGB images. The fourth column (Estimated depth) is the result obtained from LeReS which is trained on the in-domain dataset, namely Stanford2D3D. The final column (Estimated normal) is obtained by a pre-trained surface normal estimator [2, 3] tested on the generated RGB images.

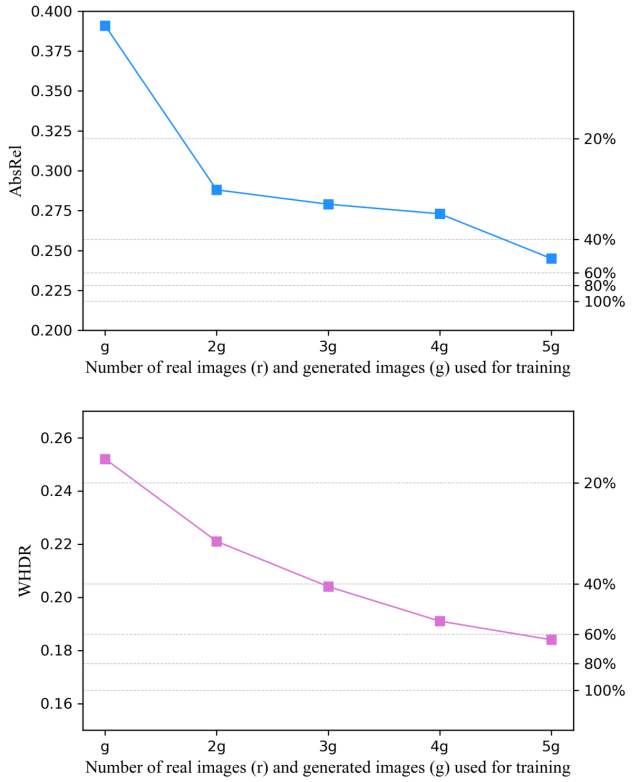

Figure 2. Depth estimation performance of LeReS [11] when trained on different number of generated data. The x-axis represents different portions of generated images. The left y-axes are the numbers of AbsRel or WHDR obtained by the corresponding models, respectively. The dashed horizontal lines indicate the performances of training on different portions of training dataset. "g" represent the numbers of generated images. In this case, $g = 12,045$, which is 20% size of the training set.

D.4. Using Only Generated Data for Depth Estimation

In addition to the results of using both generated and real data (20%) to train the depth estimation network [11], we also experiment on using generated data only. As shown in Fig. 2, using more generated data can get a steady improvement over both metrics. When the size of generated data equals to that of real data ($5g$=100% real training data), the performance is close to using 60% of real training data. This promising result indicates the possibility of using our model to generate datasets for learning downstream tasks.

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 1

[2] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *CVPR*, 2021. 2, 3

[3] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *CVPR*, 2022. 2, 3

[4] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 1

[5] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020. 1

[6] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021. 1

[7] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 1

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015. 1

[9] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? 2018. 1

[10] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1

[11] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *CVPR*, 2021. 2, 3, 4