Sim2Real 3D Scene Reconstruction

Sena Korkut Emin Sadikhov Zhenzhang Ye Nikita Araslanov
Computer Vision Group — Technical University of Munich
{sena.korkut, emin.sadikhov, nikita.araslanov}@tum.de yez@in.tum.de

Abstract

This research investigates the generalization of neural 3D reconstruction models across synthetic and real-world datasets. Specifically, we examine the gap between these domains using a model called Splatter-Image for 3D reconstruction from a single view input. Two models were evaluated: a pre-trained model and a fine-tuned model on a synthetic dataset. While the fine-tuned model outperformed the pre-trained one on the synthetic test set due to the fine-tuned model's inclinations towards generating objects with more depth, no improvements were observed on a real-world indoor scene dataset. Although our metrics suggested improvements for the fine-tuned model on synthetic data, the visual results did not fully support the metrics. We analyzed these models on real-world data, and our analysis confirmed the representation differences between synthetic and real-world datasets, highlighting the challenge of the domain gap. The difficulties of using real-world data are clearly demonstrated in this work. We provided insights into the limitations of sim-to-real generalization in 3D object reconstruction and suggest possible future directions to improve model generalization.

1. Introduction

The generalization of neural networks across different datasets is a critical and ongoing research challenge in computer vision, particularly in the domain of 3D reconstruction. While neural networks have achieved state-of-the-art results in 3D scene reconstruction, their performance can be limited by the differences between synthetic and real-world data, a problem known as the "sim-to-real" gap. Addressing this gap is important for various applications, where models trained in simulations need to work well in real-world settings. Synthetic datasets are increasingly used for training due to their ability to generate multiple consistent views with easy control over environmental variables. However, the transition from synthetic training environments to real-world application scenarios remains as an obstacle.

The sim-to-real gap exists for several key reasons. First, synthetic datasets, provide a highly structured, noise-free training environment where object models and scenes are generated in a controlled manner. In contrast, real-world datasets, capture indoor environments that are messy and unpredictable, with varying lighting, objects blocking each other, and a mix of textures and materials. These unpredictable factors in real-world environments make it harder for models trained on synthetic data to perform well. Although synthetic data offers a good base for training, it does not prepare models for the complexities of real-world environments.

To tackle these challenges, recent research has explored ways to improve how models generalize between synthetic and real-world data. For example, methods like ContraNeRF [10] use advanced learning techniques to help models handle data from different domains better. Training models directly on synthetic 3D scenes and using special methods to adapt them to real-world environments can also help bridge the gap. However, more work is needed to understand how effective these techniques are for 3D reconstruction tasks.

In this research, we demonstrate these ongoing challenges in making 3D reconstruction models work well across synthetic and real-world data. We investigate the core question of how well neural 3D reconstruction models trained on synthetic data can generalize to real-world datasets. Specifically, we use the Splatter-Image model [7], which is pre-trained on the Objaverse synthetic dataset [1], as our baseline model. We fine-tune this model using the 3D-Front synthetic dataset [2] and then evaluate its performance on real-world data from the ScanNet++ dataset [11]. Our goal is to assess how well the model, fine-tuned on synthetic data, can handle both synthetic and real-world data. We also wanted to investigate this gap with the practices of ContraNeRF, however, the required modules in their codebase were missing. This is why, we focused only on Splatter-Image. While models trained on synthetic data have demonstrated success within controlled environments, their ability to handle complex, noisy, and unstructured data from real-world scenes presents various challenges.

The rest of this paper is structured as follows: In Sec-

tion 2, we review related work on neural 3D reconstruction and sim-to-real transfer techniques. Section 3 presents the datasets and experimental setup, detailing how we prepared synthetic and real-world data for training and evaluation. Section 4 discusses the results obtained from our experiments, comparing the performance of pre-trained and fine-tuned models, both numerically and visually. Finally, Section 5 concludes with a summary of our findings and potential directions for future research.

2. Related Work

In this section, we introduce key concepts related to neural 3D reconstruction models, sim-to-real transfer, pre-training on synthetic datasets, and cross-domain techniques. These concepts provide the background for understanding the challenges and progress in improving how neural networks generalize between synthetic and real-world datasets.

2.1. Neural 3D Reconstruction Models

Neural 3D reconstruction models are designed to create detailed three-dimensional structures from two-dimensional images or incomplete 3D data. Early methods used volume-based or implicit representations [5]. More recent approaches use points or surfaces to handle larger and more detailed scenes. One notable model is Neural Radiance Fields (NeRF) [6], which can create high-quality 3D scenes from a few images. These models have greatly improved the ability of neural networks to produce realistic and detailed 3D reconstructions.

2.2. Gaussian Splatting

Gaussian Splatting [4, 12] represents a 3D scene as a mixture of G small 3D Gaussians. Each Gaussian has a mean $\mu_i \in \mathbb{R}^3$, a shape described by the covariance $\Sigma_i \in \mathbb{R}^{3\times3}$, an opacity $\sigma_i \in [0,1]$, and a color $c_i(\nu) \in \mathbb{R}^3$. The radiance field is defined as:

$$\sigma(x) = \sum_{i=1}^{G} \sigma_i g_i(x), \quad c(x,\nu) = \frac{\sum_{i=1}^{G} c_i(\nu)\sigma_i g_i(x)}{\sum_{j=1}^{G} \sigma_j g_j(x)},$$

where $g_i(x)$ is the Gaussian function. The scene is rendered by tracing rays through the Gaussians, making the process fast and efficient.

Gaussian Splatting is useful for representing 3D scenes with many small objects. It uses less memory than methods like voxels or meshes and allows faster rendering. It is efficient for large scenes that contain high-details in neural 3D reconstruction.

2.3. Splatter Image

Splatter Image [7] builds on Gaussian Splatting to efficiently reconstruct 3D objects from a single image. It predicts a 3D Gaussian per pixel from a 2D input image using a neural network. Each pixel $u = (u_1, u_2)$ has a corresponding Gaussian parameterized by opacity σ_u, mean $\mu_u \in \mathbb{R}^3$, covariance Σ_u, and color c_u. The mean μ_u is projected from 2D image space to 3D using a predicted depth d_u and offsets $\Delta_u = (\Delta_x, \Delta_y, \Delta_z)$:

$$\mu_u = \begin{bmatrix} u_1 \cdot d_u + \Delta_x \\ u_2 \cdot d_u + \Delta_y \\ d_u + \Delta_z \end{bmatrix}.$$

The model is trained using a reconstruction loss over multi-view datasets. Given a source image I, the model predicts a set of Gaussians $\theta = S(I)$, which are then rendered to form a novel view $R(\theta, \pi)$. The reconstruction loss is defined as:

$$L(S) = \frac{1}{|D|} \sum_{(I,J,\pi)\in D} \|J - R(S(I), \pi)\|^2,$$

where J is the target image, and π is the viewpoint change between the source and target cameras. This loss helps the model predict Gaussians that can accurately recreate new views of the scene with single or few-view inputs.

2.4. Synthetic-to-real Generalization

Synthetic-to-real generalization aims to train models using synthetic data while ensuring they perform well on real-world data. This is important in applications where collecting dense 3D real-world data is expensive, such as in autonomous driving, robotics, and drone navigation [8]. In the context of 3D reconstruction, generalization from synthetic to real data has been a challenge due to the domain gap between synthetic datasets, which tend to have cleaner and more consistent features, and real-world datasets, which contain noise, motion blur, and lighting variations [10]. Previous works on NeRF [6], like IBRNet [9] and GeoNeRF [3], largely focus on scene generalization using real data, often neglecting the complexities of synthetic-to-real transfer. However, experiments show that models trained on synthetic data tend to produce sharper but less accurate volume densities, resulting in more artifacts when applied to real scenes [10]. Recent methods such as ContraNeRF introduce contrastive learning with geometry constraints to bridge this gap and improve the model's performance on real data while benefiting from the detail-rich synthetic data [10].

2.5. ContraNeRF

ContraNeRF [10], different from Gaussian splatting methods, improves NeRF-based models for generating new views of scenes, especially when transitioning from synthetic to real-world data. Traditional NeRF models struggle with this transition due to differences in textures, lighting,

and noise between synthetic and real data. ContraNeRF addresses this using contrastive learning, helping the model perform better across both domains. The method achieves state-of-the-art results by training the model to recognize consistent features across different viewpoints (positive examples) while ignoring those that vary (negative examples).

By learning consistent features across views, the model understands the scene's geometry better, avoiding visual errors and ensuring that its interpretation of structure and space remains stable. It also handles real-world imperfections by focusing on geometric shapes rather than surface details, making it more reliable for real-world applications.

2.6. Pre-training on Synthetic Datasets

Pre-training models on synthetic datasets is a common way to use large amounts of labeled data without the high cost of collecting real-world data. Datasets like Objaverse [1] and 3D-Front [2] offer a wide variety of 3D models that help neural networks learn basic features. These pre-trained models can then be fine-tuned on real datasets to improve their performance. However, how well this works depends on how similar the synthetic data is to real data. If there are big differences, it can be harder to transfer what the model has learned, as discussed in studies on synthetic-to-real generalization [10].

3. Experiments

In this section, we explain the steps taken to investigate the gap between synthetic and real-world data for 3D reconstruction tasks. We used two datasets: 3D-Front [2], a synthetic dataset with indoor scenes, and ScanNet++ [11], a real-world dataset of indoor environments. First, we preprocessed the datasets to extract objects and create multiple views. Then, we fine-tuned a pre-trained Splatter Image model using the 3D-Front dataset. After fine-tuning, we evaluated the model using the ScanNet++ dataset to test how well it could perform on real-world data. The goal of this process is to better understand the model's ability to transition from synthetic to real-world data and to find the challenges in 3D reconstruction tasks caused by this gap.

3.1. Datasets

3.1.1 3D-Front

We used the 3D-Front dataset to fine-tune Splatter-Image model. 3D-Front is a large synthetic dataset with 3D room layouts filled with various furniture and objects [2]. This dataset is very useful for tasks related to understanding 3D scenes, as it contains high-quality 3D models of rooms and furniture, along with their positions and textures. Each object and room is designed to look like real-world scenes.

Our focus was on object-based reconstruction, so instead of using entire rooms, our aim was to extract individual objects like furniture from the dataset. Originally, we wanted to separate these objects from the full room scenes, keeping the noise from the scenes, such as occlusions or changes in lighting. This noise is important because, in real-world situations, objects are often partially hidden by other items or look different depending on lighting and camera angle. Including this noise would help the model perform better on real-world data, where such imperfections are common.

However, we couldn't get the camera pose data needed for the full scenes, so we had to adjust our approach. Instead of using the entire scenes, we worked with the 3D object models provided in the dataset.

3.1.2 ScanNet++

For the test set, we used ScanNet++, a real-world dataset designed for indoor scene understanding [11]. This dataset consists of detailed 3D reconstructions of various indoor environments and numerous RGB images captured from different viewpoints within each scene. These images are aligned with their corresponding 3D geometry, making the dataset particularly valuable for tasks that require a combination of 2D and 3D data.

In addition to the 3D scene data, ScanNet++ includes images taken from an iPhone camera video, which are captured from multiple angles and provide comprehensive scene coverage. Each image in the dataset is paired with its corresponding camera pose data, meaning that for every frame, the exact position and orientation of the camera about the 3D scene is available. This feature allows for accurate alignment between the 2D images and the 3D model of the room.

The availability of both the images and the camera poses makes ScanNet++ ideal for our task, which involves segmenting objects from the scene while preserving the noisy conditions that naturally occur in real-world environments. These conditions include occlusions, varying lighting conditions, and other imperfections that are present when capturing images from different angles. By using ScanNet++, we can accurately simulate these real-world challenges and ensure that our model is exposed to a wide range of visual noise.

3.2. Preprocessing

For the preprocessing of the 3D-Front dataset, we took 3D object models that are used for constructing 3D scenes. These objects are mainly furniture. To fine-tune the model, we needed multiple views of each object and the corresponding camera poses. We generated these views by randomly rotating the objects and creating 128×128 images from each view. We also saved the camera's transformation matrix for each view, so the model could learn from different angles of the objects.

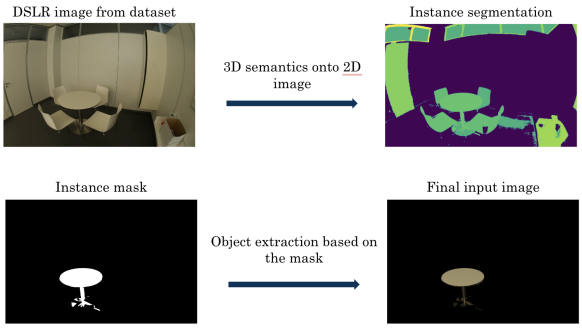

Figure 1. The process of extracting object instances from DSLR images. The final input image contains the masked object, preserving scene noise and occlusions.

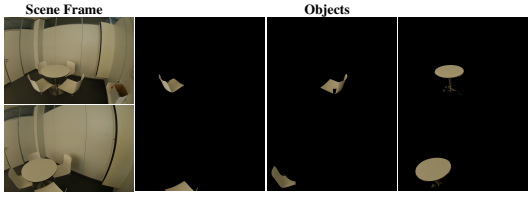

Table 1. Scene frames and corresponding objects extracted from that frame. The instances match for the two example scenes, and represents the same object from different views.

For the test dataset, we required multiple views of the object along with their corresponding camera poses. To achieve this, we began by utilizing DSLR images from the dataset. While the dataset provides semantic and instance segmentation information for the 3D scenes, it does not provide such information for the 2D images directly. To bridge this gap, we projected the 3D semantic data onto the 2D images by making use of the available camera pose information.

By rasterizing the 3D semantics onto the 2D image, we were able to generate instance segmentation for the 2D images. Each pixel in the 2D image was assigned a specific instance ID, allowing us to generate a mask for each object instance easily. This mask is to segment individual objects from the scene. In Figure 1, the mask for the table is shown, and based on this mask, we were able to extract the object from the scene.

We repeated this process for five different DSLR images within each scene, which enabled us to capture multiple views of the same object from different angles. The scene frames were taken from a video, and in some frames, completely different pieces of furniture appeared because the camera view was from the opposite side of the room, exposing completely different objects. To avoid this inconsistency and be able to select and extract objects automatically, we only selected a portion of the frames from the start of the video, where there were only small changes in camera pose between each frame. As a result, we did not obtain multiple views of objects from drastically different perspectives like we did with 3D-Front objects, but rather from slightly varied angles. Table 1 demonstrates examples of different views for one scene, and how the final objects are masked out from those frames.

To determine which objects to extract, we prioritized the ones based on the top instance labels provided by the ScanNet++ dataset. The objects we focused on for extraction were chairs, tables, office chairs, cabinets, bookshelves, sofas, beds, monitors, storage cabinets, and doors. This ensured that the most relevant and available objects were selected for segmentation and further processing. Through this method, we achieved the required object views and corresponding camera poses for our test dataset.

3.3. Model Setup

For the model benchmarking, we utilized two models. The first model is a pre-trained version provided by the authors of the Splatter-Image technique. This model was originally trained on the Objaverse dataset, a comprehensive and large-scale dataset containing a wide variety of object types [1].

To adapt this pre-trained model to our specific task, we fine-tuned it using the 3D-Front dataset, which is more focused on indoor scene furniture. Additionally, we attempted to train a model from scratch using only the 3D-Front dataset. However, due to the relatively small size of the 3D-Front dataset, this approach did not yield any meaningful results, and as a result, we chose to discard the model trained from scratch.

3.4. Evaluation

For evaluation, we followed the standards of Splatter-Image [7]. We evaluate the quality of our 3D reconstructions using three key metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). PSNR looks at the pixel-level differences between the original and reconstructed images, measuring how much an image has been corrupted by noise. SSIM focuses on the structural similarity of the images, comparing things like brightness, contrast, and how well the structure is preserved. A higher SSIM score shows that the reconstruction keeps the important structural details of the original image. LPIPS is a metric that uses neural networks to measure how similar two images appear to a person. Lower LPIPS values mean the reconstructed image looks more like the original in terms of human perception.

To evaluate the models, Splatter-Image inputs a single image that is preprocessed before being fed. The preprocessing involves extracting the object from the background, normalizing its position, and adding a white background to

Dataset	Model	PSNR ↑	SSIM ↑	LPIPS ↓
3DFront	Pre-trained	18.906	0.873	0.136
	Fine-tuned	**23.439**	**0.919**	**0.121**
Scannet++	Pre-trained	2.843	0.230	0.449
	Fine-tuned	**3.445**	**0.259**	**0.442**

Table 2. The results show the performance of both pre-trained and fine-tuned models, with higher PSNR and SSIM values indicating better reconstruction quality and lower LPIPS values indicating improved perceptual similarity.

the image. If the image size differs from 128×128, it is reshaped accordingly. We followed this same preprocessing method for our evaluations.

4. Results

The results in Table 2 show that the fine-tuned model performs better than the pre-trained model when evaluated on the 3D-Front dataset. The fine-tuned model performs better in terms of PSNR and SSIM, meaning that it produces more accurate and structurally similar reconstructions. Additionally, the LPIPS score, is lower for the fine-tuned model, indicating higher quality. This improvement is likely because the fine-tuned model was trained on the 3D-Front dataset. Since the model had already seen similar data during training, it was better at handling the test set, leading to stronger results. The model learned the structure and characteristics of 3D-Front objects during training, which helped it perform better when tested on the same type of data.

The views created for the evaluation of 3D reconstruction in 3D-Front are created by rotating the 3D object mesh randomly. This means the evaluation is likely to calculate all reconstructed parts of the object, including the front, back, sides, and even the underside. The fine-tuned model has a more complete understanding of the object's structure, which explains the improved performance.

However, when we compare the pre-trained and fine-tuned models on the ScanNet++ dataset, the comparison is not as reliable or meaningful. The views for ScanNet++ were generated using video captures, as explained in Section 3.2. These video captures provide only slightly different perspectives of the object in each frame. This means that most of the evaluation views are very similar to the input image, showing mostly the front of the object. Because of this, the calculations does not get to see the back, underside, or other important parts of the object that would help it evaluate the reconstruction of the object fully.

As a result, the evaluation on ScanNet++ only tests how well the model can reconstruct the front of the object and does not give a full picture of the model's abilities. This is why the performance numbers on ScanNet++ are not truly representative of how well the model works, and they don't give a clear comparison between the pre-trained and fine-tuned models. The limited views make it difficult to see how well the model can generalize and reconstruct the whole object, so the numbers in the table for ScanNet++ don't tell the full story about the differences between the two models.

During the process of generating visual results, we noticed a difference between the 3D objects produced by the pre-trained and fine-tuned models. The objects generated by the pre-trained model appeared more compact and well-defined, while the objects from the fine-tuned model extended further in depth and became less clear the deeper they went. This pattern was consistent across different input images, as shown in Table 3.

What was surprising, though, was how different the visual results were compared to the numerical results. Based on the metrics, we expected the fine-tuned model to produce reconstructions that were at least similar or slightly more accurate than those from the pre-trained model. However, the visual outputs showed much larger differences between the two models than we had predicted. We thought the fine-tuned model would generate clearer reconstructions, but it instead produced more depth and non-clear predictions in some areas, making the results appear less sharp.

For the visual results to appear non-clear and exaggerated in depth, one possible explanation could be the nature of the 3D-Front dataset, which was used to fine-tune the model. This dataset contains larger objects, such as sofas and beds, which may have led the fine-tuned model to prioritize generating objects with more depth and volume. The model may have adapted to these large-scale objects by creating deeper structures, even if that introduces some lack of sharpness in the final output. This tendency to emphasize depth could explain why the fine-tuned model produces 3D reconstructions that appear less compact and more extended compared to the pre-trained model.

The numerical results may be slightly better with the fine-tuned model because it tends to create more depth in its reconstructions. When evaluating 3D-Front objects, which often have significant depth, this extra focus on depth can lead to more predictions in that area, potentially improving the model's performance metrics. This increased depth may help compensate for the numerical difference between the fine-tuned and pre-trained versions. However, this does not necessarily mean that the fine-tuned model is better overall. As we observed in the visual results, the fine-tuned model often produces unclear predictions in the depth regions, leading to reconstructions that lack meaningful structure.

When evaluating the models with the ScanNet++ dataset, we noticed that the input images were quite different from those in the 3D-Front dataset. This is largely because Scan-

Input Image	Pre-trained	Finetuned
		
		
		
		
		
		
		
		
		

Table 3. Reconstructed 3D objects generated by pre-trained and fine-tuned models. Each row corresponds to an input image of a furniture item (table, chair, and sofa) selected from 3D-Front dataset, with the results shown from three different views.

Input Image	Pre-trained	Finetuned
		
		
		
		
		
		
		
		
		

Table 4. Reconstructed 3D objects generated by pre-trained and fine-tuned models. Each row corresponds to an input image of a furniture item (bookshelf, table, and chair) selected from ScanNet++ dataset, with the results shown from three different views.

Net++ is a real-world dataset, capturing more variability and noise in the scenes. As seen in Table 4, the input images are also noisier. For instance, in the case of a table and bookshelf, there are objects placed on it, but since we are extracting only the table object based on its instance segmentation, the items on top are excluded from the image. This exclusion is consistent across multiple views of the same object, resulting in an object representation with holes or missing parts. Additionally, not all views contain the complete object; in some cases, the scene only captures part of the object, cutting off important sections. This inconsistency makes it harder for the model to fully understand and

Dataset	Model	PSNR ↑	SSIM ↑	LPIPS ↓
Scannet++	Pretrained	2.571	0.236	**0.438**
	Fine-tuned	**2.975**	**0.253**	0.441

Table 5. The results show the performance of both pre-trained and fine-tuned models specifically calculated for intuitively difficult objects.

reconstruct the object from the available data, leading to incomplete representations.

Such representations make it difficult to interpret the numerical results accurately. Furthermore, both models attempt to reconstruct these areas of missing data, but instead of leaving the holes, they fill them in with white, which is the background color. This happens because we follow the same pre-processing method described in the Splatter-Image paper, where the background of the input image is replaced with white before being fed into the model. Having random holes on the object complicates the reconstruction process for the model, making it harder to generate the object's full structure accurately. This factor further highlights the challenges of working with real-world data, where noise, incomplete object information, and inconsistent views introduce additional difficulties for 3D reconstruction.

4.1. Intuitively Difficult Objects

After observing that the fine-tuned model tends to generate results with more depth, we became curious about how the model would handle the reconstruction of objects that might be particularly challenging for a neural network to learn. Specifically, we focused on objects like curtains, blankets, and blinds, which are inherently difficult to model in 3D. These objects often have varying shapes depending on their position in a scene. For instance, a blanket can take on many different 3D forms depending on how it is placed, and each blanket might have a unique shape, making it more complex for a model to capture compared to more rigid, stable objects like beds or sofas. This variability poses a challenge for the model to generalize across different instances of the same type of object. As a result, we decided to investigate how well the model performs in reconstructing these difficult objects to better understand its limitations in handling complex shapes.

Table 5 presents the numerical results for intuitively difficult objects, where the metrics are computed solely based on specific labels and their reconstruction quality. These results are similar to those discussed previously, showing only slight differences.

Instead of numerical results, the visual results in Table 6 provide a clearer demonstration of the difference between

Table 6. Reconstructed 3D objects generated by pre-trained and fine-tuned models. The left column displays the original input scenes and extracted objects (blanket and curtain) from ScanNet++ dataset, with the results shown from three different views for each model.

the two models. The fine-tuned model produces reconstructions that are less opaque and exhibit greater depth, providing a more detailed 3D representation. In contrast, the pre-trained model generates more compact reconstructions, with less emphasis on the third dimension, which may actually be advantageous for certain objects like curtains. This difference is not fully captured in the numerical results, as the evaluation is based on limited viewpoints—primarily slight variations of the front side of the object, as mentioned before. This limitation prevents the numerical metrics from reflecting the full spatial quality and depth that the visual results show.

5. Conclusion

In this paper, we studied how well neural 3D reconstruction models can work on both synthetic and real-world datasets, focusing on the gap between them. We used the Splatter Image as a base model, pre-trained on the Objaverse dataset, and fine-tuned it on 3D-Front synthetic dataset. We evaluated both models on both synthetic and

real-world data to understand the challenges of generalization from synthetic to real-world environments. Our experiments showed a noticeable difference in performance when moving from synthetic to real-world data, and the challenges of what is known as the "sim-to-real gap".

Our results showed that the fine-tuned model performed better on synthetic dataset than the real-world dataset, Scannet++. This improvement could be seen in higher scores in metrics like PSNR, SSIM, and LPIPS. The models were better at producing accurate and clear 3D reconstructions after being trained on data similar to the test set. However, when the models were tested on the real-world ScanNet++ dataset, the results were not meaningful. Both models had trouble creating high-quality reconstructions for real-world data, showing that models trained on synthetic datasets could not generalize well.

A challenge we demonstrated is that synthetic datasets are helpful for training but they do not capture the complexity of real-world environments. Synthetic scenes are usually well-organized, free from noise, and consistent in lighting and object placement. In contrast, real-world scenes, like those in ScanNet++, are messy, with lighting changes, occlusions, and random object arrangements. This makes it hard for models trained on synthetic data to perform well in the real world.

We also noted some limitations in the way we evaluated the models. Metrics like PSNR, SSIM, and LPIPS, while useful, do not fully capture the quality of 3D reconstructions when there is a difference in the views we capture for the objects. The visual results showed that the fine-tuned model often produces 3D reconstructions with more depth, but these reconstructions are sometimes less clear and noisier. On the other hand, the pre-trained model makes more compact and simpler reconstructions. These differences aren't always visible in the numerical results because we had to test the models on only front views of the objects for ScanNet++ dataset. This is also one of the limitations of real-world data, where it is hard to obtain desired inputs for a proper evaluation.

The gap between synthetic and real-world datasets shows the difficulty of model generalization in synthetic-to-real cases. One future work idea is to train models directly on 3D scenes, instead of focusing on individual objects. By learning from whole scenes, the models could better understand the relationships between objects, lighting, and occlusions, which are critical for performing well in real-world environments. This approach could help the models capture the complexities of scenes for better generalization across domains. Ideas like integrating contrastive learning and adjusting the loss function to focus on general object features instead of small details could also possibly improve the generalization of models trained with synthetic datasets when evaluated on real-world datasets.

References

[1] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022. 1, 3, 4

[2] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Jiaming Wang Cao Li, Zengqi Xun, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3d-front: 3d furnished rooms with layouts and semantics, 2021. 1, 3

[3] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors, 2022. 2

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 2

[5] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[6] 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, 2020. 2

[7] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 4

[8] Joanne Truong, Sonia Chernova, and Dhruv Batra. Bidirectional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters*, 6(2):2634–2641, Apr. 2021. 2

[9] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering, 2021. 2

[10] Hao Yang, Lanqing Hong, Aoxue Li, Tianyang Hu, Zhenguo Li, Gim Hee Lee, and Liwei Wang. Contranerf: Generalizable neural radiance fields for synthetic-to-real novel view synthesis via contrastive learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 3

[11] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 1, 3

[12] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. 2