IGSR: Iterative Gaussian Splatting Refinement for Limited-View Scenarios

Yash Taneja
Stanford University
Aeronautics & Astronautics
ytaneja@stanford.edu

Abstract

This project introduces IGSR, an iterative refinement scheme to enhance Gaussian Splatting models from limited-view datasets. While existing methods have solved this problem of 3D reconstruction from limited viewpoints, they rely upon heavy computational requirements and large models. I propose a novel method that chains the most efficient modern models together to improve efficiency across the entire pipeline of 3D scene reconstruction on a single 16GB NVIDIA T4 GPU. The developments from this project are three-fold: (1) a custom and simple denoiser model designed to reduce noise characteristic of poor Gaussian Splatting models, (2) an iterative process that incorporates this denoiser to augment the dataset with new views that extend the periphery of initial dataset, and (3) experimentation that compares the effect of initialization and data selection for augmentation, with synthetic and real data. The results indicate that the denoiser model, as trained, is generalizing to new scenes and even real data, but it is suboptimally denoising unseen views for the IGSR pipeline. The methods show promise, but the denoiser architecture served to be a weak-link in the process.

1. Introduction and Related Work

Modeling a 3D scene from limited views is useful across many applications such as robotic navigation, low radiation dosage medical imaging, surgical planning, and historical scene reconstruction. However, as expected, this is a difficult task that often results in poor quality results. This project aims to investigate an iterative approach to improve the quality of limited-view 3D reconstruction using state of the art methods including DUSt3R [1] to preprocess images for 3D reconstruction, Gaussian Splatting [2] as the method of 3D reconstruction to synthesize novel views, and a supervised learning pipeline for denoising unseen views.

Gaussian Splatting [2] is a method used to represent a complete 3D scene explicitly with a set of 3D Gaussians, constructed from a finite number of input views. The ultimate task that this accomplishes is novel-view synthesis, or being able to extract unseen views from limited data, which has been used in a wide variety of computer vision tasks across applications [3]. Gaussian Splatting is not the only way to achieve this task–beyond traditional explicit representations such as voxel grids, pointclouds, and meshes, a recently popular implicit representation is through Neural Radiance Fields (NeRFs) [4]. NeRFs employ a multi-layer perceptron representation of the scene, but unlike Gaussian Splatting, NeRFs generally face a tradeoff between efficiency and quality since they rely upon a neural representation.

Each Gaussian that composes a Gaussian Splatting model is parameterized by a three-dimensional mean μ and covariance Σ matrix (Eqn. 1), opacity, and a set of spherical harmonic coefficients (to capture the view-dependent color of real scenes). Given points from a Structure from Motion (SfM) [5] pipeline, the Gaussians go through a fully differentiable process to fit the inputted 2D images onto 3D space. There is also an adaptive density control segment that selectively adds or prunes Gaussians based on a heuristic [2].

$$G(x) = \frac{1}{(2\pi)^{3/2}\sqrt{|\Sigma|}} \exp -\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu) \quad (1)$$

COLMAP [5] is the SfM preprocessing pipeline used in vanilla Gaussian Splatting, to estimate camera poses and generate a 3D point cloud from 2D images. This process relies on a series of independent engineering problems including feature extraction, point correspondence matching, triangulation, and bundle adjustment, which have each been researched extensively and optimized. However, none of these steps can be performed with consistent perfection (for example, the bundle adjustment step relies on an estimate of camera intrinsics, which are difficult to gauge exactly), and hence errors propagate throughout the entire SfM process. DUSt3R, or Dense Unconstrained Stereo 3D Reconstruction [1], was created to address this issue. In DUSt3R, all of the steps used in COLMAP are combined in an end-to-end transformer-based framework that can consistently generate accurate 3D point reconstructions from any set of sparse images. For the task of multi-view depth estimation, the best DUSt3R results are obtained over 1000x faster than COLMAP [1].

This DUSt3R process does not rely on camera intrinsics and generally relaxes the hard requirements involved in traditional pose estimation. In their framework, images are handled in a pairwise manner, where each image of the pair is fed through an identical ViT encoder. Then, the latent representations are passed through two transformer decoders which share information with cross-attention (this is the primary mechanism to achieve correspondances between the images). From this, there is a regression head to output (1) a pointmap $\in \mathbb{R}^{W \times H \times 3}$ for each image, that maps image pixels to a 3D position, and (2) confidence $\in \mathbb{R}^{W \times H}$ for each image, that associates the confidence of the network's 3D position mapping for each pixel. The pointmap is generated in the coordinate frame of the first image. The encoder used in DUSt3R relies upon a pre-trained CroCo ViT [6], which uses Masked Image Modeling (MIM) as its self-supervision mechanism. MIM is particularly suited to generate effective features in DUSt3R's downstream reconstruction task since it helps encode the context information necessary to ultimately identify correspondances within pairs of images in the decoder step.

In InstantSplat [7], DUSt3R is incorporated to replace the COLMAP preprocessing step for generating Gaussian Splatting models from pose-free and sparse-view image data. Using DUSt3R further enables several optimizations to the overall Gaussian Splatting process, such as eliminating the complicated adaptive density control heuristic and implementing a faster optimization scheme that leverages the richer point-cloud models from DUSt3R (as opposed to sparse models from COLMAP). With these optimizations, the paper impressively cites that the process takes under one minute on an Nvidia A100 GPU. This indicates that DUSt3R can be used as an effective preprocessing step for limited view Gaussian Splatting.

Note that despite the DUSt3R preprocessing step, we can still expect a subpar 3D reconstruction when the input viewpoints are too sparse and points are predicted with low confidence values. Hence, this is an active area of research. In GANeRF [8], the authors propose an adversarial framework by incorporating a discriminator to encourage the NeRF to encode a better 3D representation in order to produce more realistic views that do not contain artificial artifacts. This work is mainly intended to increase NeRF quality and was not necessarily designed for limited-viewpoint situations; still, as described in the Methods section, my project derives inspiration from GANeRF by experimenting with a GAN-based denoiser enhanced with an LPIPS loss [9], [10], [11]. Moreover, in ReconFusion [12], the authors intend to achieve a similar task of creating better NeRF representations, but by leveraging diffusion models instead of GAN frameworks. In ReconFusion, the authors specifically investigate limited-view situations and demonstrate successful reconstructions with as few as three viewpoints. Finally, in ZeroNVS [13], Sargent et al. demonstrate 360-degree view synthesis from a *single image* during inference, also leveraging generative diffusion models. This is the current state-of-the-art.

As described, there are existing state-of-the-art methods for modeling a 3D scene from limited views. However, they are exceptionally data intensive, have long training times, and are computationally expensive. For this project, I use a compact 16GB RAM Nvidia T4 GPU, and I focus on developing a simple denoiser to achieve limited-view 3D reconstruction. This project proposes a strategy that sequentially augments an initial limited-view dataset with denoised unseen views from a Gaussian Splatting model. Throughout the proposed pipeline, I have experimented with the current state-of-the-art methods in terms of efficiency including DUSt3R for the SfM preprocessing step, Gaussian Splatting for the 3D reconstruction, and a simple supervised learning framework for denoising. The overall pipeline can be visualized in Figure 1 and is further described in the Methods section.

2. Methods

2.1. Iterative Gaussian Splatting Refinement

As discussed, this project aims to investigate methods to increase the accuracy of 3D Gaussian Splatting models from limited-viewpoint image inputs. In the Iterative Gaussian Splatting Refinement (IGSR) pipeline, the inputs are raw images of a single object from limited viewpoints (Dataset described below). These inputs get pre-processed through DUSt3R [1] for camera pose estimation and sparse 3D point-cloud generation, producing an output identical in form to what COLMAP [5] traditionally creates. This will then serve as the input for Gaussian Splatting [2] to create an initial model of the entire scene, from which we will derive unseen images. The process so far is adapted from [14].

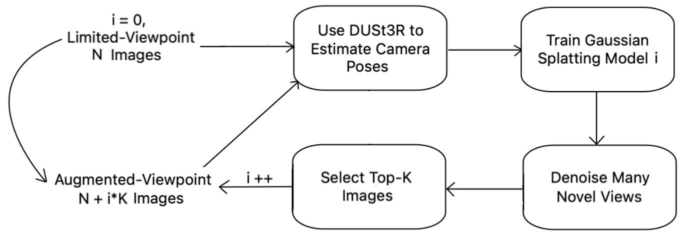


Figure 1. Pipeline for the proposed Iterative Gaussian Splatting Refinement (IGSR) process.

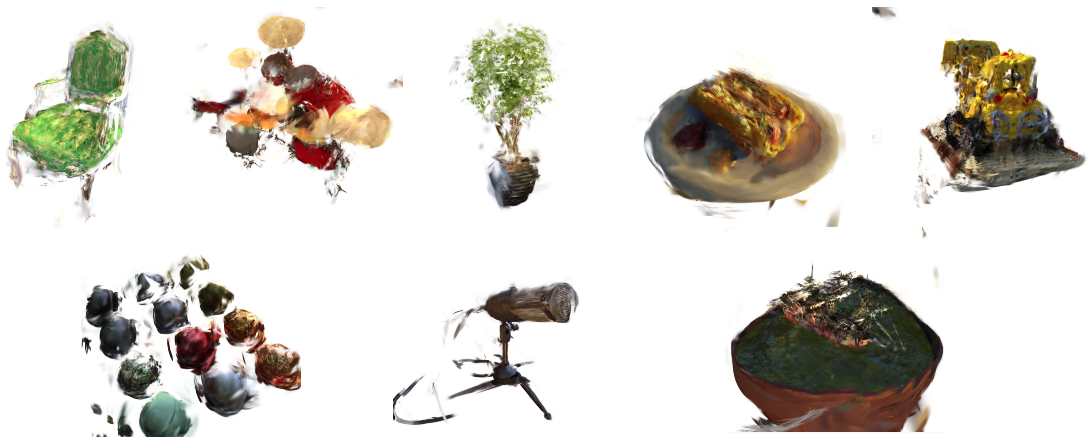


Figure 2. Representative synthetically-noisy images from all 8 scenes from the NeRF Synthetic dataset including ”chair,” ”drums,” ”ficus,” ”hotdog,” ”lego,” ”materials,” ”mic,” and ”ship.” The former four scenes form the training data, the ”lego” scene forms the validation data, and the latter three scenes form the test data.

At this stage, the model is expected to be a poor representation of the scene, since it was initialized with limited viewpoints. From here, we can extract many views from different poses across the Gaussian Splatting model, including those in areas we have little information about the scene. These images go through a denoising process (described below), and the Top-K most realistic images are selected to augment the initial pool of limited-view images. From here, the process is repeated as many times as desired until a Gaussian Splatting model that adequately represents the full 3D scene is created. It is predicted that the quality of the Gaussian Splatting model will be improved each time this pipeline is repeated, as measured by peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS) metrics between Gaussian Splatting generated views and ground truth views (these metrics are standard for image quality assessment). This pipeline is summarized in Figure 1.

Regarding the Top-K selection of the most realistic images after denoising, I use the LPIPS metric, which relies upon access to ground truth images at each denoised pose. This is suboptimal as elaborated in the Future Work section, but it is sufficient for the aim of this project to demonstrate efficacy of the methods. After evaluating LPIPS metrics, there are two options for how we choose the Top-K images. *Top-K Improvement* chooses the images that show the greatest decrease in LPIPS after denoising, corresponding to the greatest increase in realism. *Top-K Absolute* chooses the images that have the lowest LPIPS value overall, corresponding to the images with the most realism, either pre- or post-denoising. Evaluation of these two options is described in the Experiments section.

Moreover, throughout the pipeline, a single 16GB NVIDIA T4 GPU is utilized. Gaussian Splatting [2] is performed with 5000 iterations for every model created. DUSt3R [1] was limited by memory, only being able to only process 15 images at a time. This is due to the use of a heavy-duty transformer backbone and because images are fed in pairs (all pair combinations are tested), as described in the Introduction. Note that COLMAP [5] did not provide effective pose estimates since it is difficult to find correspondences across sparse images. DUSt3R is able to produce good pose estimates for reasons mentioned in the Introduction, and it employs a confidence threshold that was tuned in [14] to mask out image regions with poor confidence.

So far, the denoising model is described as a black box. We cannot describe the denoising model until we describe the dataset.

2.2. Dataset for Denoising

The primary dataset that this project relies on is a NeRF Synthetic Dataset from [4] that contains 8 synthetically-generated scenes with 300 800x800 images per scene. There was significant preprocessing required before this dataset was ready to train the denoiser. Since I wanted to deploy a supervised learning framework, I has to generate noisy images with associated ground truth label images. The final result of this preprocessing effort can be seen in Figure 2.

It was important that the particular ”noise” applied to the images was representative of the noise to be used in the downstream task of Gaussian Splatting refinement. Hence, I first ran a three-view DUSt3R-to-Gaussian-Splatting-model pipeline (similar to what was described above), synthesized novel views, and understood what ”noise” looks like in a poorly rendered Gaussian Splatting model, as seen in the left image of Figure 3. There were also, as expected, completely garbage views in areas of the scene that the three-view input did not cover, but I chose to ignore these areas, since it would be difficult to train a good-enough generator to hallucinate accurate representations of these areas of the scene with the limited data that I have decided to work with. This is acceptable since the denoiser is proposed to be used in an iterative manner, where we can selectively use it in areas that are on the periphery of the known view, and ultimately grow it with each iteration. The effectiveness of the denoiser is evaluated in the Experiments section below, where I investigate the effect of a scattered vs localized limited-view initialization.

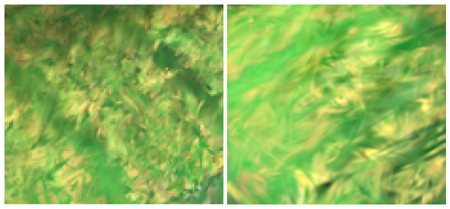


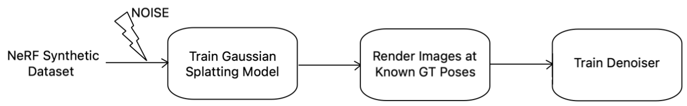
Figure 3. Zoomed-in noisy view from a poor limited-view Gaussian Splatting model (left) and a similar synthetic-noise view (right) of the "chair" scene.

Hence, knowing what the noisy images should roughly look like, I found that the synthetic nature of the dataset being used lent itself well to manual noising. That is, I simply chose a subset of camera poses (20 out of 200) and added Gaussian noise ($\mu = 0, \sigma = 0.01$) to all the transform matrices. Then, after training for 5000 iterations upon the 20 images corresponding to the noised poses, I rendered unseen views (which had associated ground-truth poses) for each scene, and found that the noise looks qualitatively representative of noise in a poorly-initialized Gaussian Splatting model as shown in Figure 3, and I was even able to generate random artifacts in the scene. Note that some views were noisier than others based on their closeness to the 20 initial poses, and there was a good diversity in the images (see Figure 2). I predicted that this will be useful in avoiding overfitting when we train the denoiser.

In arriving at this, notice that there are several parameters: noise σ, number of camera poses, specific camera poses chosen (scattered or localized), and number of iterations to train the Gaussian Splatting model. I recognized that the desired noise could be generated in two ways: (1) if we train with many scattered images and some noise, but for a short number of iterations, or (2) with a few images and some noise, but for a longer number of iterations. The former led to blurry images, but not noisy in the way that was desired. Hence, the latter paradigm was selected and the parameters described in the previous paragraph were found to be the most representative.

For each scene (8), were 300 noisy images generated, with associated ground-truth label images. Five scenes were used for training, one scene for validation, and three scenes for testing, as described in Figure 2.

This overall dataset preprocessing pipeline is summarized in Figure 4.


NOISE
NeRF Synthetic Dataset
Train Gaussian Splatting Model
Render Images at Known GT Poses
Train Denoiser
Figure 4. Pipeline for dataset preprocessing to train the denoiser.

2.3. Denoising Model

Now that I have described the data used to train the denoising model, I can provide details behind the architecture. Qualitatively, the goal of the denoiser is to "know" what the noise looks like in a Gaussian Splatting model, remove it, and fill in the gaps by generating an image that not only looks real, but also agrees with the other images in the dataset and the pose from which it was acquired. Initially, I was exploring existing super-resolution models that are intended to increase the quality of low-resolution images, for fine-tuning purposes. However, I decided not to go down this path for two reasons: (1) these were extremely large models, so it went against my overall goal to develop a low-computation solution, and (2) after running an image from my dataset through the super-resolution model, I realized that the idea of super-resolution does not align with the idea of denoising for Gaussian Splatting models since it is not built to *add* structure to the model, only modify it.

Hence, I next explored architectures to train from scratch. I recognized that I did not have a large enough dataset to train a highly generalizable denoiser, but it would serve as a proof-of-concept in my overall IGSR pipeline. The first model I trained was a Generative Adversarial Network (GAN) [9] architecture, since we would be able to leverage the adversarial nature of a generator/discriminator framework to encourage the generator to produce images that have "realness." The hope was that with enough data, the generator would be able to hallucinate structures in the image that are occluded by noise. I used the vanilla generator and discriminator loss functions from [9] as shown below, where z is the noisy image and x is the ground truth image at that same pose.

$$l_G = -\mathbb{E}_{z\ p_{data}}[\log D(G(z))] \tag{2}$$

$$l_D = -\mathbb{E}_{x\ p_{data}}[\log D(x)] - \mathbb{E}_{z\ p(z)}[\log(1 - D(G(z)))] \tag{3}$$

Noticing that the resulting colors from the GAN were completely off, I incorporated a perceptual loss by adding an LPIPS loss [10] to the generator loss function.

$$l_G = -(1-a)\mathbb{E}_{z\ p_{data}}[\log D(G(z))] + aLPIPS(z, x) \tag{4}$$

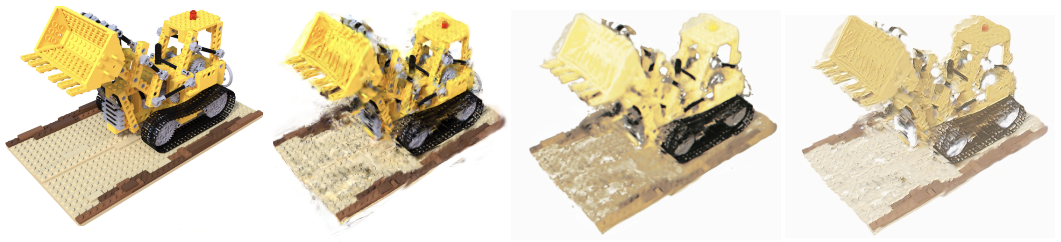


Figure 5. Ground truth (top), GAN denoised output (middle), and LPIPS network denoised output (bottom).

Here, a is a hyperparameter that weighs the two losses accordingly, and $LPIPS(z, x)$ is defined below.

$$LPIPS(z,x) = \sum_{l} \alpha_l ||\hat{F}_l(z) - \hat{F}_l(x)||_2^2 \qquad (5)$$

The goal of LPIPS is to understand the similarity of two images in a deep network's feature representation. In Equation 5, $\hat{F}$ is the normalized feature representation of an image in layer l and α_l are the learned weights of a pre-trained deep network (several deep network backbones are compatible, including VGG [15] and AlexNet [16]).

From initial experimentation when I was trying to overfit a small subset of data (300 noisy and ground truth images from the ”lego” scene), I noticed that the colors only qualitatively looked right when the hyperparameter a was set near 1, meaning that the network is primarily learning through just the LPIPS loss, not the adversarial mechanism of the GAN. I wanted to conduct a study between using a purely LPIPS loss network vs a GAN/LPIPS loss network with $a = 0.9$, so I trained the model for 5 epochs on the ”lego” scene, with a batch size of 4 (this was the maximum that my single 16GB T4 GPU could support).

Qualitative results are shown in Figure 5. Not only is the pure-LPIPS loss producing qualitatively closer colors to the ground truth, but the structural details and general denoising capabilities are also better than that from the GAN/LPIPS loss. In addition, the training loss is extremely sporadic in the GAN-based network compared to the stable, converging pure LPIPS network (specific training loss curves are provided Appendix Figure 8 if the reader is interested). Hence, I decided to move forward with training the pure LPIPS network.

I selected hyperparameters using data from four scenes (”chair,” ”drums,” ”ficus,” and ”hotdog”) as the training set, and one scene (”lego”) as the validation set. I performed several experiments with different hyperparameters. With the Adam optimizer [17], a learning-rate of 1e-3 with betas 0.9 and 0.999 yielded the best validation loss. For the LPIPS loss function itself, the VGG backbone instead of the AlexNet backbone yielded the best validation loss. A total of 2 epochs for training yielded a very good validation loss, with diminishing improvements in loss with further epochs (see Appendix Figure 9 for the training and validation loss of the final network). Since my dataset is not very large, I was worried about overfitting, so I erred on the side of choosing the lowest possible number of epochs (2 epochs), while still achieving low loss.

3. Experiments, Results, and Discussion

3.1. Denoising Model Evaluation

To characterize the performance of the denoiser, after being trained as described in the previous section, I evaluated PSNR, SSIM, and LPIPS metrics on all available data before and after denoising, as summarized in Table 1. Note that average results from the scenes processed in InstantSplat [7] are provided as a baseline for what really good Gaussian Splatting results should look like. If the denoiser was perfect, the post-denoising metrics should be in the same ballpark as the InstantSplat results, but we can see that this is not the case. These quantitative results capture high-level information across all scenes. Qualitative images to visualize the denoiser outputs are provided in Appendix Figures 10 and 11 if the reader is interested.

As expected, since the denoiser was trained on an LPIPS metric, the LPIPS value decreased for all scenes, except the ”ship” test scene. Low LPIPS value is associated with high image realism, correlating with human visual judgement [10]. Similarly, SSIM increased for all scenes, except the ”ship” test scene. Like LPIPS, SSIM is also good at measuring human-like visual similarity across images because it assigns high scores to images that have similar structure, and structure has been found to be the most important to humans when assessing the similarity of images. Since LPIPS improved, it is not surprising that SSIM improved too. Finally, PSNR consistently decreased across all cases. PSNR is a direct, pixel-wise comparison metric that is not well-suited for human visual realism, but it is effective at capturing the fine-grained similarity of pixel values between images. As seen in the Appendix Figures, the denoised images tend to have a faded tint about them, explaining the worsening of PSNR values.

Through this analysis, we can clearly see that the denoiser is not achieving its goal of filtering out the noise of

Table 1. PSNR, SSIM, and LPIPS metrics evaluated on all 300 images from all 8 scenes before and after denoising (compared with known ground truth images).

	Scene	Before Denoising			After Denoising		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Train	Chair	18.867	0.814	0.187	17.284	0.832	0.176
	Drums	14.078	0.727	0.311	13.301	0.773	0.261
	Ficus	17.757	0.829	0.160	16.525	0.845	0.138
	Hot Dog	18.694	0.829	0.260	15.114	0.831	0.224
Val	Lego	16.258	0.716	0.281	13.610	0.731	0.258
Test	Materials	12.579	0.722	0.295	11.249	0.751	0.261
	Mic	16.103	0.860	0.160	14.859	0.874	0.148
	Ship	13.107	0.634	0.401	10.468	0.613	0.412
InstantSplat [7]	Avg. Results	28.58	0.89	0.13	—	—	—

renders from a poor Gaussian Splatting model. Our denoising model has learned how to improve LPIPS, but it does so in a way that does not actually increase realism or accuracy compared to ground truth. This is a limitation of the denoiser model, and potential modifications are discussed in the Future Work section. Still, it is evident that the model, as trained, does generalize across scenes, and is consistently improving LPIPS, even for scenes that were not included in the training set. This is with the exception of the ”ship” scene, which is likely because the $\sigma = 0.01$ Gaussian noise applied to all scenes may have been too much for this particular scene (this can be seen in the noisy ”ship” image in Appendix Figure 11).

Overall, I will use this denoiser model as-is for future experiments, since I can still use it to characterize other aspects of IGSR, while recognizing that the denoiser is suboptimal.

3.2. Limited-View Initialization

One aspect of the IGSR pipeline that is worth investigating is the effect of initialization upon the overall results. That is, for the initial Gaussian Splatting model ($i = 0$ in Figure 1) whose renders are ultimately denoised for the iterative process, I investigate whether a localized or scattered initialization is more effective. In addition, this experiment has an additional variable of using Top-2 Improvement vs Top-2 Absolute selection criteria for dataset augmentation (as described in the IGSR Methods section).

A localized initialization would be one where all images are from a similar viewpoint/camera pose (this was determined qualitatively for the purpose of this experiment). This type of initialization would ultimately lend itself best towards single-view Gaussian Splatting. A practical example of this type of initialization is in a single-robot navigation scenario, where the robot may only be able to capture images from a limited viewpoint and would like to perform state estimation and navigation within a Gaussian Splatting environment. On the other hand, a scattered initialization is one where all images are from completely different viewpoints/camera poses (this was also determined qualitatively for this experiment). This type of initialization is expected to perform better within IGSR since there would be a better representation of the overall scene. A practical example of this type of initialization is in a scenario where there are multiple, but sparse cameras to capture a snapshot of a changing environment.

For this experiment, I manually held-out a single evaluation dataset of 15 images that represented a wide-variety of viewpoints across the scene. Metrics were computed for camera poses corresponding to these 15 images as found through DUSt3R [1] for consistency (recall that 15 images was the max that DUSt3R could support without running out of memory). Since we identified several shortcomings of the denoiser in the previous section, I began with a scene that was directly used for training the denoiser (”chair” scene), I used 10 images from this scene as initialization, and I only ran one iteration, for the best chance of making IGSR work. If positive results were recovered, I would proceed to evaluate test scenes and experiment with different numbers of iterations and initial images. The overall pipeline, the two selected images (the two small box pairs in the figure) from the Top-2 Improvement and Top-2 Absolute criteria, and qualitative results of an arbitrary ground truth pose from the evaluation set, is summarized in Figure 6. Quantitative results containing PSNR, SSIM, and LPIPS are shown in Table 2. Note that Iteration 1.1 and 1.2 correspond to the Top-2 Improvement and Top-2 Absolute selection criteria of unseen views for data augmentation, respectively (these unseen views are different views than the held-out evaluation set).

From Table 2, we can see that the only improvement in metrics after an iteration are for PSNR and SSIM for localized initialization, using the Top-2 Improvement selection criteria. However, these values are poor compared to those in InstantSplat. Also, the LPIPS metric worsened after an iteration of IGSR. Even though the denoiser is known to effectively improve LPIPS for a given image as found in the previous section, it did not contribute to an improvement in LPIPS of the overall evaluation in the framework of IGSR. As discussed in the previous section, high PSNR

Table 2. PSNR, SSIM, and LPIPS metrics for the "chair" scene, evaluated on a subset of 15 held-out ground truth images, for a single iteration. Iteration 1.1 and 1.2 correspond to using the Top-2 Improvement and Top-2 Absolute selection criteria, respectively, for dataset augmentation of the next iteration. This table corresponds to the quantitative data associated with Figure 6.

	Iteration 0			Iteration 1.1			Iteration 1.2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Localized Initialization	11.259	0.567	0.427	12.555	0.590	0.448	11.140	0.561	0.431
Scattered Initialization	12.523	0.752	0.312	12.402	0.731	0.343	12.236	0.727	0.344
InstantSplat [7] (avg. results)	28.58	0.89	0.13	—	—	—	—	—	—

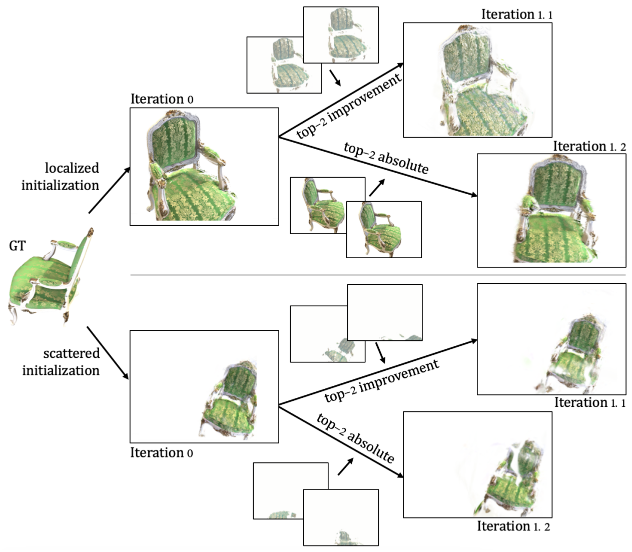


Figure 6. IGSR with localized initialization vs scattered initialization of 10 images, for 1 iteration. Top-2 Improvement and Top-2 Absolute selection are compared for dataset augmentation (selected images are shown in small boxes). Note that the selected ground truth image is from the evaluation set, independent of the Top-2 selection criteria.

indicates strong correlation with the ground truth evaluation poses at the pixel-level, and high SSIM indicates a similar image structure to the ground truth. However, from Figure 6, we can see that neither of these is the case for any image after an iteration of IGSR, and it can be deduced that the increase in PSNR and SSIM values for localized initialization and Top-2 Improvement selection criteria for data augmentation was a matter of coincidence, where the faded tint effect happened to help these metrics. For this reason, the Top-2 Absolute criteria is actually preferred, since it enables data augmentation with the best possible unseen rendered images (pre- or post-denoising). Top-2 Improvement criteria could only work better if the denoiser was accurate to begin with.

Contrary to expectations, the scattered initialization performed more poorly than the localized initialization, both quantitatively and qualitatively. In fact, for the chosen ground truth pose in Figure 6, it is apparent that the render from the initial Gaussian Splatting model that uses the scattered initialization is completely wrong, and even out of frame. This indicates that despite having better coverage of the entire scene, DUSt3R had a hard time with extracting camera poses due to the sparse nature of the inputs. As a result, the Top-2 selected poses from both criteria (shown as the small images in Figure 6) were meaningless after being denoised, since they were out of view and out of focus to begin with. For this reason, localized initialization is preferred over scattered initialization.

Overall, a localized initialization and the use of the Top-2 Absolute criteria is preferred. Note that the localized initialization case enjoyed the best LPIPS value overall for an unseen image (pre-denoising) because DUSt3R was able to generate higher confidence camera poses of the localized region, leading to a more accurate Gaussian Splatting representation in that region, and a higher chance that we could augment the data with a good render (ideally on the periphery of our localized region). However, if not carefully done, IGSR could propagate errors across iterations, especially since the Gaussian Splatting model was observed to be extremely poor and noisy in regions that we did not have an initialization image. With localized initialization, the model does not know how to wrap *around* an object, and it instead appears to have a warping/flattening effect. (With scattered initialization, the model's unseen poses were just out-of-view or out-of-focus.)

3.3. Real Data Iteration

Since positive results were not recovered in the previous experiment, further experimentation with number of iterations and initialization images was not performed for the synthetic scenes. To produce an additional datapoint, I decided to not only test with fewer initialization images and an additional iteration, but I also tested with real images of an Apple AirPod taken on my iPhone 12. This experiment is set up in the exact same way as the previous experiment, taking 4 localized images for initialization, with 7 scattered held-out evaluation images and 6 scattered "unseen" images (we extract unseen images at the pose associated with these images). Top-1 Absolute is used as the selection criteria for data augmentation, and for both iterations, this additional image comes from the pre-denoised set.

Qualitative results are provided in Figure 7 (notice there are white patches in the image because of the DUSt3R [1] confidence thresholding described previously). Renders

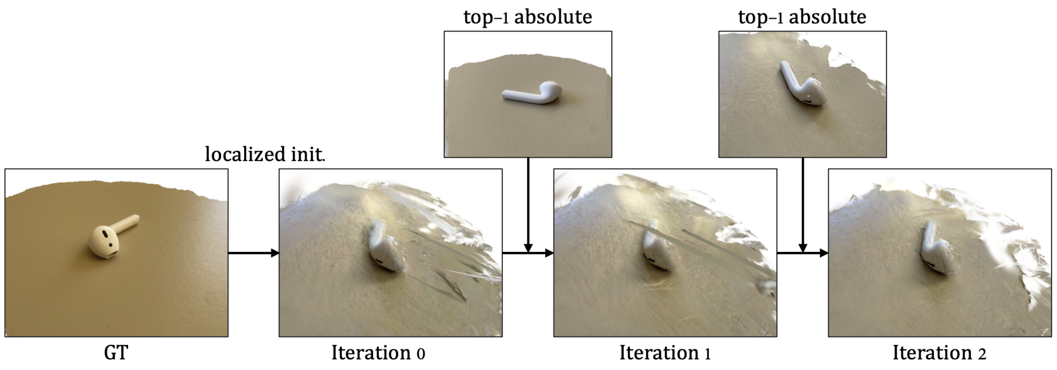


Figure 7. IGSR with localized initialization of 4 images for two iterations with real images of an Apple AirPod, taken on an iPhone 12 camera. Top-1 Absolute selection criteria was used for dataset augmentation for each iteration.

Table 3. PSNR, SSIM, and LPIPS metrics for a real data scene of an Apple AirPod, evaluated on a subset of 7 held-out images, for two iterations. The Top-2 Absolute selection criteria was used throughout the process for dataset augmentation of the subsequent iteration.

	Iteration 0			Iteration 1			Iteration 2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Real Scene	15.448	0.707	0.573	15.216	0.724	0.575	14.195	0.677	0.597
InstantSplat [7] (avg. results)	28.58	0.89	0.13	—	—	—	—	—	—

from the arbitrary ground truth image in the evaluation set are shown at each iteration, and we can see that the renders are at the wrong pose, though they do appear to become slightly denoised with each iteration. As discussed with the previous experiment, it appears this incorrect pose error is propagated through IGSR iterations, though it is promising how the images become less noisy over time. Quantitative results are provided in Table 3, where all metrics are moving adversely across iterations, except for there being a brief benefit in SSIM from at iteration 1. This is likely attributed to the fact that the selected Top-1 Absolute image being in the same distribution as the localized initialization images.

4. Conclusion and Future Work

In this project, I have designed an Iterative Gaussian Splatting Refinement (IGSR) scheme that incorporates a simple denoiser trained with an LPIPS loss upon a custom dataset that models noise in a poorly initialized Gaussian Splatting model. Experimentation demonstrates that a localized initialization and the Top-K Absolute selection criteria (selecting the an unseen view with the best LPIPS value pre- or post-denoising for data augmentation in the next iteration) are preferred for IGSR, though error propagation effects are observed when applied on real and synthetic datasets.

In the future, since the weakest link is the denoiser, I have many ideas to experiment with: incorporate a pixel-wise loss during training since PSNR was the only metric that degraded after denoising, backprop through a few layers of the LPIPS deep network backbone, test regularization in the loss function or through dropout, use self-attention so that the denoiser can attend over the entire batch so that the generated image is self-consistent with the dataset. Though unrelated to this specific project, when I was determining how to construct the dataset, I noticed that training with an abundance of images, but for only a short period of time, resulted in a blurry dataset; this blur is alleviated with more training, but perhaps using a super-resolution model instead of longer training could result in faster overall training.

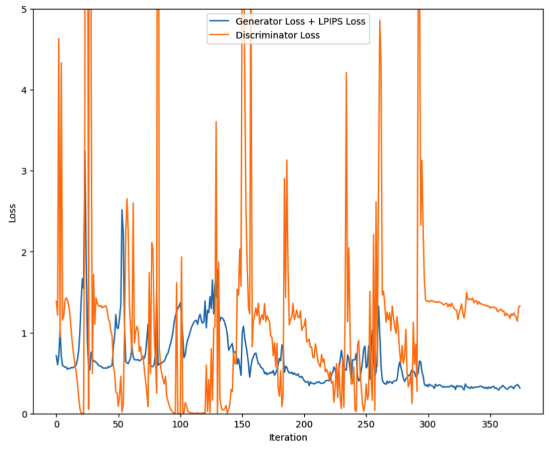
Finally, there are some important future considerations before bringing IGSR to fruition. First, as noted in the scattered initialization experiment, DUSt3R does not perform as well, and it leads to faulty camera pose estimation. I fear that the evaluation dataset camera poses may also have suffered from this effect, so I would like to try generating an evaluation dataset with significantly more data to verify my results. Since DUSt3R is GPU limited, this would require a stronger GPU. Second, the current Top-K method relies upon having access to ground truth images for metric evaluation—this is not deployable in the wild, and we could instead modify the selection criteria to be by closeness to existing poses, since IGSR will work best by extending the periphery of localized regions, as discussed. Third, I am interested to explore how the denoiser works at scale—maybe after nailing down a performant denoiser, we could constructing a foundation model level Gaussian Splatting denoiser, potentially leveraging diffusion models in a way similar to [13] or [18]. Finally, I would like to test on more scenes in the wild, perhaps using the Temples and Tanks dataset [19] used in InstantSplat [7], or even extending to dynamic scenes.

5. Contributions and Acknowledgements

I would like to acknowledge Professor Mac Schwager and Dr. Ola Shorinwa from the Multi-Robot Systems Lab at Stanford for their advice and brainstorming sessions during the ideation phase of this project at the start of the quarter. In addition, I would like to thank all the CS231N teaching assistants for their support throughout the quarter, with a special appreciation for Samir Agarwala for his mentorship and productive ideation sessions, Ishikaa Lunawat for introducing me to DUSt3R and helping me get started on the project, Lucas Leanza for his empathy and debugging support when my Google Cloud VM was not cooperating, and Kyle Sargent for pointing me towards related papers (including his own) and helping me debug the denoiser when it was completely malfunctioning.

6. Appendix

6.1. Loss Plots



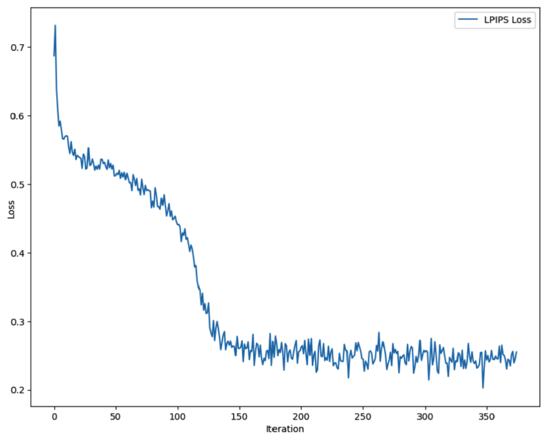


Figure 8. For the case of overfitting the "lego" scene for 5 epochs, the top plot shows the GAN Training losses, with blue representing the generator + LPIPS loss and orange representing the discriminator loss. The bottom plot represents the pure LPIPS network training loss.

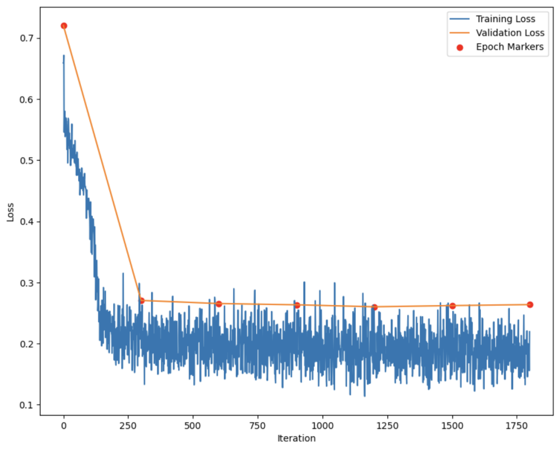


Figure 9. Final training loss and validation loss of the LPIPS network after hyperparameter tuning. The validation loss was only calculated once per epoch. The batch size was 4 and the number of training examples was 1200.

6.2. Qualitative Images for Denoising Experiment

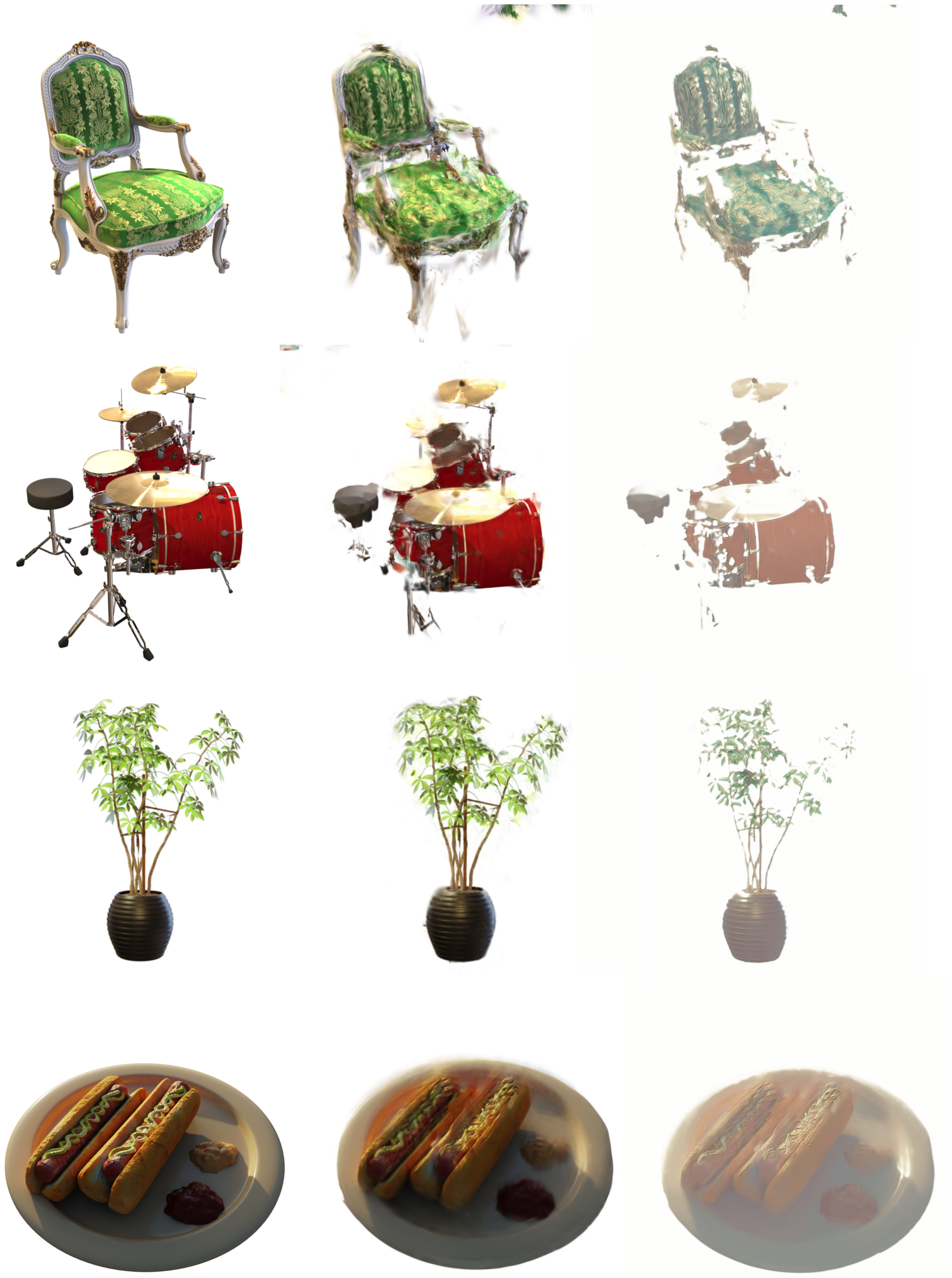


Figure 10. Ground truth (left), noised (middle), and denoised (right) images for ”chair,” ”drums,” ”ficus,” and ”hotdog” scenes. These scenes were used for training the denoiser.

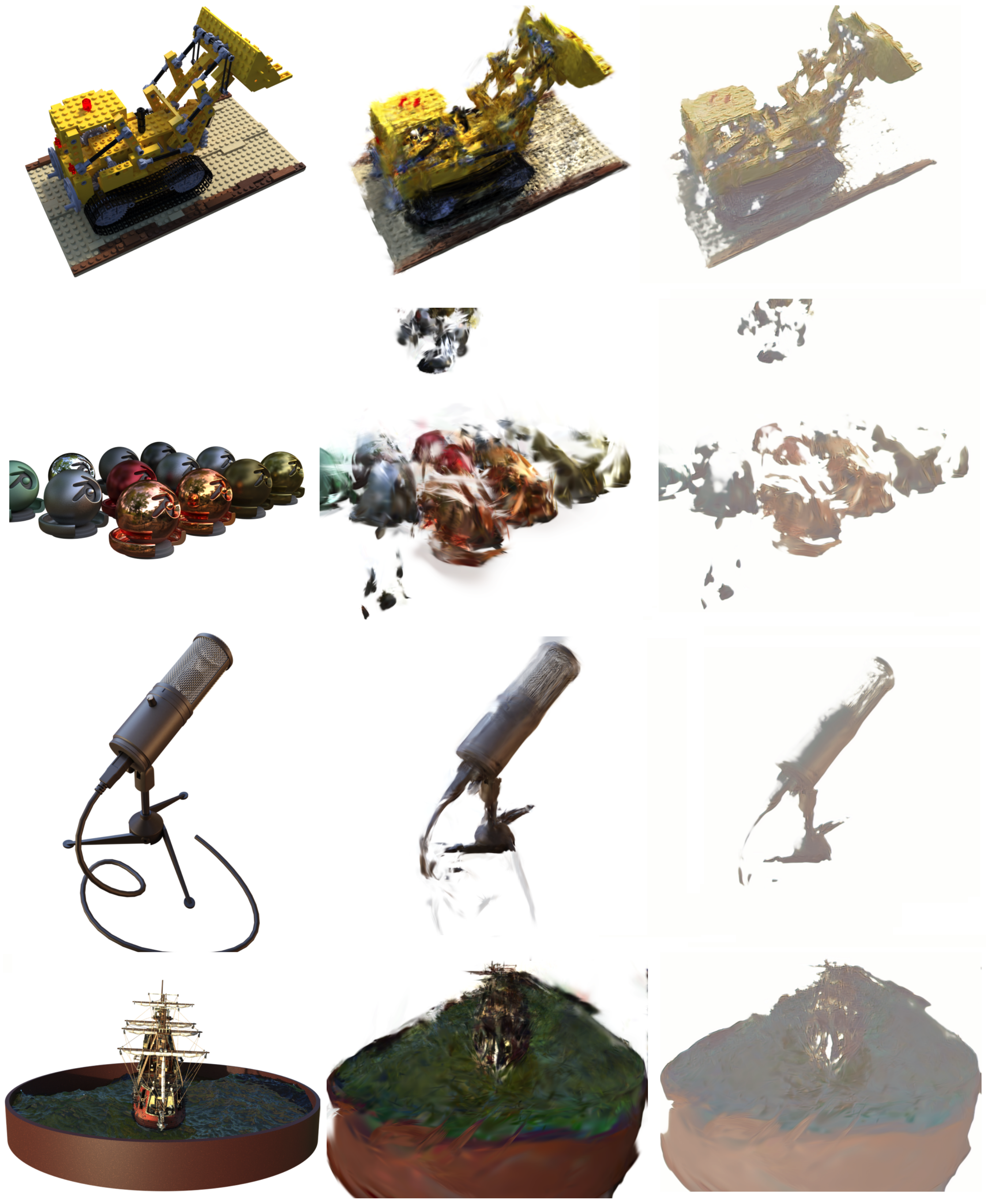


Figure 11. Ground truth (left), noised (middle), and denoised (right) images for ”lego,” ”materials,” ”mic,” and ”ship” scenes. These scenes were not used for training the denoiser (though the ”lego” scene was used as a validation scene for hyperparameter tuning).

References

[1] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy, 2023.

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023.

[3] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting, 2024.

[4] 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. *CoRR*, abs/2003.08934, 2020.

[5] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion, 2023.

[7] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds, 2024.

[8] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Niessner. Ganerf: Leveraging discriminators to optimize neural radiance fields. *ACM Transactions on Graphics*, 42(6):1–14, December 2023.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[10] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[11] Younghyun Jo, Sejong Yang, and Seon Joo Kim. Investigating loss functions for extreme super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 424–425, 2020.

[12] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Holynski. Reconfusion: 3d reconstruction with diffusion priors, 2023.

[13] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, and Jiajun Wu. Zeronvs: Zero-shot 360-degree view synthesis from a single image, 2024.

[14] Kovalenko Daniel. Wild-gaussian-splatting. `https://github.com/nerlfield/wild-gaussian-splatting`, 2024.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[18] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

[19] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.