Improving novel view synthesis of 3D Gaussian splats using 2D image enhancement methods

Wouter Bant
University of Amsterdam
wouter.bant@student.uva.nl

Ádám Divák
University of Amsterdam
adam.divak@student.uva.nl

Jasper Eppink
University of Amsterdam
jasper.eppink@student.uva.nl

Clio Feng
University of Amsterdam
clio.feng@student.uva.nl

Roos Hutter
University of Amsterdam
roos.hutter@student.uva.nl

Abstract

Novel view reconstruction based on only 2 input images is an important but extremely challenging task. pixelSplat is a potential solution that was shown to deliver high-quality results at a competitive speed. We first evaluate pixelSplat on more challenging reconstruction tasks by applying camera positions that are further away from each other, and find that its performance is heavily impacted. We then explore 2D image enhancement methods to fix the corrupted novel view images. A diffusion model-based solution proves to be able to restore significantly impacted areas, but fails to stay consistent with the original scene even after long fine-tuning, resulting in flickering videos. An alternative solution based on an image restoration model results in pleasant videos and quantitative improvements in most metrics, but does not address all errors seen in the novel view images. We explore the underlying reasons for these shortcomings, and propose future research directions for fixing them. Our code and videos are publicly available on GitHub.[1]

1. Introduction

In this paper, we explore a pipeline designed to enhance the performance of novel view synthesis from sparse image observations. Novel view synthesis is important in a wide range of downstream tasks, but in our case, we focus on generating videos of home interiors based on only two input images, which could be used for example in home advertisements. While great progress has been made recently on multi-view consistent novel view generation for single objects [28], replicating this for complete scenes based on two images remains a challenging task.

[1] https://github.com/WouterBant/diffusion-augmented-pixelsplat

Novel view synthesis has been a significant area of research for many years. One of the early promising techniques in this field is multi-view stereo (MVS) [12, 14, 23, 31]. MVS techniques generate dense 3D representations by leveraging multiple images taken from different viewpoints. They typically build upon Structure from Motion (SfM) [36]. This combination of SfM and MVS allows for creating highly accurate and detailed 3D reconstructions.

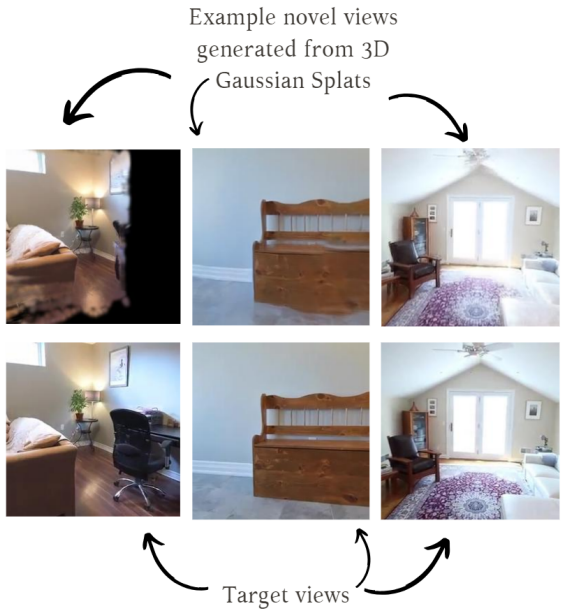

Figure 1. Examples where Gaussian Splatting generates poor novel views because of unobserved parts of the scene. The three main types of errors are omissions, inconsistencies, and degraded image quality.

Recently, neural techniques have advanced significantly with the introduction of Neural Radiance Field (NeRF)

methods [29, 30, 39]. NeRF methods facilitate high-quality novel view synthesis; however, one of their major drawbacks is the lengthy training time required. Training time for NeRF methods is scene-specific, causing lengthy training time to be problematic. Another promising neural approach is using light-field transformers [10, 33, 34], developed to address the extended training times associated with volume rendering techniques. Although light-field transformers offer faster processing times, they still fall short of real-time performance. A significant limitation of light-field transformers is their inability to reconstruct 3D scene representations that can be edited or exported for downstream tasks in computer vision and graphics.

To address these limitations, 3D Gaussian methods have been proposed, enabling real-time rendering with minimal memory requirements through rasterization-based volume rendering [22]. PixelSplat [3] is an extension of this technique, allowing for the generation of 3D Gaussians in a single forward pass from as few as 2 images. This approach also introduces a novel rasterization technique that enables real-time novel view synthesis. PixelSplat has been proven to generate high-quality reconstructions when the novel viewpoint is close to the original views used for reconstruction. However, we noticed a marked degradation in performance when testing on the same data set with more varied camera positions. Figure 1 shows that generating novel views can result in blurry images, black areas, or spatially inconsistent reconstructions where 3D Gaussians are underrepresented in a given area.

To mitigate these issues, we explored how incorporating prior knowledge can be used to fill in or improve badly reconstructed areas. While the best solution would be to enhance the underlying 3D representation of the scene, this is an extremely challenging task due to the lack of training data and the architecture of pixelSplat. On the other hand, there are models trained on large amounts of images that have learned visual priors over a wide range of real-world scenarios. This is why we propose a pipeline where the potentially corrupted novel view output of pixelSplat is improved using image enhancement models. We test multiple ways of performing image enhancement, using both off-the-shelf and fine-tuned model versions. We train and evaluate on the RealEstate10k dataset and results are compared to the standard pixelSplat output in terms of novel view image quality, inference speed, and temporal consistency.

2. Related work

Traditional Scene Reconstruction. Early novel view synthesis methods were initially based on light fields, starting with densely sampled approaches [16, 26], then progressing to unstructured capture [2]. The introduction of Structure-from-Motion (SfM) [35] opened up a new domain where collections of photos could synthesize novel views. SfM first estimates a sparse point cloud during camera calibration, which was initially used for basic 3D visualization. Subsequent advancements in multi-view stereo (MVS) [15] led to impressive full 3D reconstruction algorithms [4, 11, 18, 25], enabling the development of various view synthesis techniques. These methods re-project and blend input images into the novel view camera, using geometry to guide the process. While these methods often produce excellent results, they struggle with unreconstructed regions or "over-reconstruction" caused by nonexistent geometry from MVS.

Neural Radiance Fields. Early approaches to novel view synthesis utilized deep learning [13, 43], with methods like CNNs estimating blending weights [18] or employing texture-space solutions. These methods often relied on MVS geometry and suffered from temporal flickering during rendering. Volumetric representations, exemplified by Soft3D, combined deep learning with volumetric ray-marching to represent geometry. Neural Radiance Fields (NeRFs) further improved quality but suffered from slow rendering due to their large Multi-Layer Perceptron (MLP). Recent advancements, like Mip-NeRF360, focused on improving both quality and speed through regularization strategies. Newer methods prioritize faster training and rendering by leveraging spatial data structures, different encodings, and MLP capacity adjustments.

3D reconstruction using 3D Gaussian Splatting. Point-based representations have been pivotal in rendering techniques, initially proposing rendering point sets as independent geometry samples. These methods have been efficiently implemented in graphics software and have been highly parallelized on GPU hardware. To address issues like holes in incomplete surfaces, the concept of "splatting" points with extents larger than a pixel has been explored. Recent advancements in this domain, such as 3D Gaussian Splatting (3DGS), have revolutionized rendering quality and speed. By introducing adaptive density control and efficient rasterization kernels, 3DGS achieves unprecedented quality and speed, making it applicable in various domains like 3D human and avatar reconstruction, 3D generation, SLAM systems, 4D reconstruction, and open-set segmentation. Efforts have also been made to address aliasing and point densification in the 3DGS representation, while recent research explores compression techniques for 3DGS, resulting in more compact scenes and faster rendering. Using 3DGS techniques allows for real-time rendering, which was impossible with the previous NeRF techniques.

2D Image enhancement models. In computer vision, degradations are undesirable effects such as noise, motion blur, haze, and low dynamic range, among others. Restoring a high-quality image from its degraded version is a complex inverse problem, as there can be multiple potential solutions for a given degraded image. Recent advances in deep learn-

ing have consistently outperformed traditional techniques in the domain of blind image restoration. State-of-the-art neural networks for this task include convolutional neural networks (CNNs) [9], Transformers [40], and Diffusion models [38]. In the image-enhancing field, models exist that are trained specifically for each different type of degradation, however, this approach is resource-intensive and less practical in scenarios where the desired enhancement is not clearly defined. Our research focuses on all-in-one restoration models, which are designed to handle various image-enhancing tasks such as denoising, inpainting, and deblurring, among others. Notable examples of such models include SwinIR [27], InstructIR [8], and ControlNet [41]. We will explore the capabilities of these models and their effectiveness in multi-task image enhancement. These versatile models aim to streamline the image restoration process, making it more efficient and adaptable to diverse degradation scenarios.

3. Background

3.1. Gaussian splatting

3D Gaussian Splatting (3DGS) [22] parametererizes a 3D scene as a set of 3D Gaussian primatives $g_k = (\mu_k, \Sigma_k, \alpha_k, S_k)$. Every Gaussian primitive k consists of 4 parts. Each gaussian has a mean $\mu_k \in \mathbb{R}^3$ which is the center point in a 3D coordinate system. $\Sigma_k \in \mathbb{R}^{3\times3}$ is the corresponding covariance matrix. $\alpha_k \in \mathbb{R}$ denotes the opacity and S_k are spherical harmonics coefficients. The collection of these primitives defines the underlying scene and can be rendered to produce novel views. These Gaussian primitives can be rendered via an inexpensive rasterization operation, allowing rendering to be real-time. The fitting of a 3DGS model is closely related to the fitting of a Gaussian mixture model, where we seek the parameters of a set of Gaussians such that the likelihood of a set of samples is maximized. One downside of this approach and a general problem is that this optimization is non-convex and typically solved with the Expectation-Maximization (EM) algorithm. This algorithm suffers from local minima [21].

3.2. PixelSplat

PixelSplat [3] adapts the standard 3DGS model by incorporating a two-view image encoder and a pixel-aligned Gaussian prediction module. The two-view image encoder is essential because different scenes C_i are scaled by individual, arbitrary scale factors $s_i \in \mathbb{R}^+$, leading to scale ambiguity between scenes. This encoder resolves the scale ambiguity by employing per-pixel correspondence via epipolar attention [17], resulting in a scale-aware feature map $F[u]$ at pixel coordinate u. Subsequently, the parameters of the Gaussian primitives are predicted. For each pixel, the depth feature is used as input to predict the parameters of M Gaussian primitives. To predict the Gaussian features for each pixel, a network is first employed $(\phi, \delta, \Sigma, S) = f(F[\mu])$. Using these predicted parameters, the Gaussian mean μ is then calculated with the camera intrinsics and the predicted offset parameter δ. Finally, to ensure differentiable sampling, they set $\alpha = \phi_z$, yielding the predicted parameters for the Gaussians in a single forward pass.

3.3. ControlNet

ControlNet [41] is used to steer the outputs of diffusion models based on an image and optional prompt. Diffusion models learn to predict the reverse process of noise addition to generate data from noise. In short, ControlNet makes a copy of the original diffusion model, the original model is kept fixed and the copy will be fine-tuned. The copy receives the prompt and input image and will based on that provide embeddings to the original model via 1x1 convolutions. This way the original model is preserved and will only follow the input image when the ControlNet conditioning is given. Intuitively ControlNet learns how to provide inputs to the original diffusion model by fine-tuning a copy of this diffusion model on the data.

3.4. InstructIR

InstructIR [8] is an all-in-one image restoration method. It is trained to fix image degradations, such as noise, motion blur, haze, and low dynamic range, based on a free-form textual prompt. InstructIR contains an image model, which is based on NAFNet [5], that itself is based on the UNet architecture [32], and a simple sentence encoder for encoding text instructions. It uses a task routing mechanism that based on a prompt and image embedding decides which task to perform as a soft combination of feature layers.

InstructIR was trained to fix 6 specific degradations based on existing data sets (denoising, deraining, dehazing, deblurring, low-light enhancement, and image enhancement). Some of these are highly similar to the quality issues we would like to fix, which is why InstructIR was selected for evaluation. We do not rely on the textual prompting capabilities of InstructIR, so it would make sense to use just the underlying NAFNet image model directly. However, the availability of a pre-trained multi-purpose model checkpoint made it easier to test InstructIR.

4. Methods

4.1. Pipeline

Our proposed pipeline is made up of several successive steps, as shown in Figure 2. First, at least two images of the same scene are selected, for which the relative camera positions are already known - these images are referred to as *context* images. Then, a novel viewpoint is selected, referred to as a *target* image, for which the 2D representation

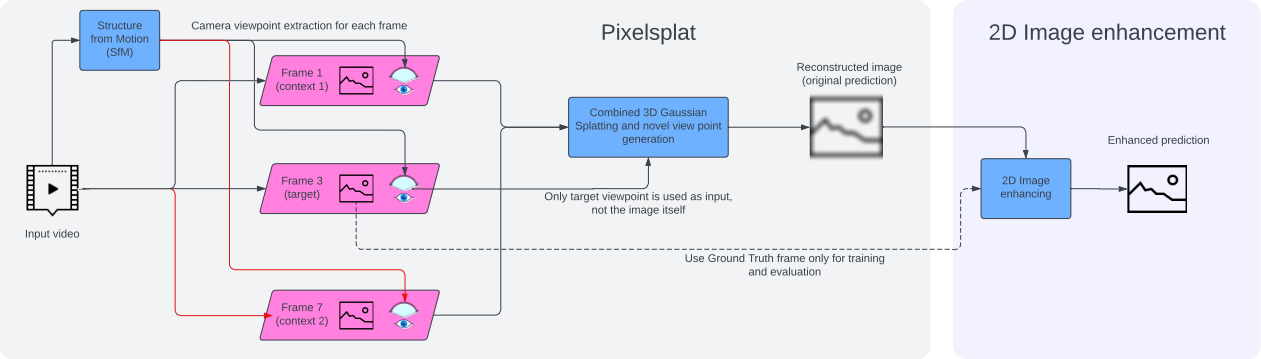

Figure 2. Main steps of the pipeline: 1. Camera viewpoints are extracted from a video using Structure from Motion, 2. pixelSplat is invoked with two *context images* and their respective viewpoints, and an additional third novel viewpoint as its input, which describes the *target image*, 3. pixelSplat generates a single image representing this novel viewpoint, referred to as the *original prediction*, 4. 2D image enhancement is applied solely on the *original prediction*, without any other input, producing the final *enhanced prediction*. During evaluation, this *enhanced prediction* is compared to the *target image*.

is also known, but this is only used for the final evaluation. The pre-trained pixelSplat model is used for generating the 2D image from this novel viewpoint based on the two *context* images and their viewpoints. We call this image *original prediction*. Next, the 2D image enhancement model is used for creating an improved version of this 2D image, referred to as *enhanced prediction*. Finally, this *enhanced prediction* image is compared to the *target* image. The image enhancement module only receives the reconstructed 2D input image, and optionally a generic prompt that is the same for all scenes, as its input - it does not use camera positions, the *context* images, previously calculated prediction images, or any scene-specific prompting.

Training is performed in two distinct stages - first, the pixelSplat model is trained on the given data set, then the trained model is used to generate reconstructed novel view images. Then these novel view images and their ground truth pairs are used to train the image enhancement module separately. This separation made it possible to re-use the pre-trained pixelSplat model provided by the original authors and to reduce the resource requirements of the project. On the other hand, it made it impossible to propagate gradients from the image enhancement module to pixelSplat during training. This could theoretically improve performance, but we believe it would not yield a significant improvement at this point.

4.2. Data

We use RealEstate10k [44], a dataset of home walkthrough videos downloaded from YouTube, for all our experiments. Having access to complete videos instead of single images is advantageous, as we can test a multitude of input and novel view combinations for each scene. The videos show realistic scenes with a mostly consistent speed of camera motion, so each frame of the video can be treated as a separate viewpoint. RealEstate10k already contains the camera viewpoint for each frame extracted using Structure from Motion, which is a prerequisite for the pixelSplat model. The authors of pixelSplat kindly shared the converted representation of the data set upon request, so it was immediately usable by their code.

The pixelSplat implementation defines the distance between the two context images and the target images as a function of the time between the frames in the input video. This makes the definition easy, but it does not give spatial guarantees, for example, there is no way to specify a minimum or maximum amount of overlap between the camera viewpoints.

The pixelSplat authors used a narrow baseline of 2 context images that are 45 frames apart in the video, and 4 target images that are anywhere between the two context images. Given the speed of camera motion in most videos in RealEstate10k, these settings resulted in viewpoints that were close to each other, which is why we will refer to these settings as the *easy* task.

In order to test a more realistic scenario, we modified the parameters to use a wider baseline of 135 frames between the context images and enforced that all target images must be at least 12 frames apart from the context image. We generated 106k train and 11.5k test images from the original train and test scenes of the RealEstate10k dataset. This *hard* task resulted in a setup that is significantly more difficult, and where the output of pixelSplat degraded heavily, which made image enhancement justified. Due to the lack of direct controls on viewpoint overlap, it did result in some scenes where the task became unintentionally difficult, as a large

part of the predicted image was missing from the context images, resulting in large black areas.

4.3. Image enhancement

There are three main types of errors in the *original prediction* generated by pixelSplat: 1. omissions; 2. inconsistencies; and 3. degraded image quality. Omissions refer to areas that are completely or largely missing (i.e. black). Inconsistencies refer to areas where the reconstruction is spatially impossible or contradictory, for example, when different pieces of furniture blend. Degraded image quality refers to areas in the image where the main outline of objects is correct, but details are missing or blurry. Samples of these errors can be seen in Figure 1.

Initially, we aimed to deliver image enhancement by an inpainting pipeline: identifying regions of the image that were degraded and only modifying the selected areas. However, this method would be most useful in correcting omission errors, but less useful in correcting the other two issues.

To tackle all three problems at the same time, we decided to use a diffusion model instead, which operates on the entire image simultaneously. When using ControlNet, the diffusion process starts with random noise and is guided by the textual prompt and a conditioning image. We experimented with using either no prompt or a uniform prompt for all scenes. We used the *original prediction* as the conditioning input and fine-tuned the model on the novel views generated from the training set of RealEstate10k, as described in Section 4.2. We experimented with increasing the image conditioning scale parameter to steer the generation to stay close to the original scene instead of generating a completely different one. Despite the fine-tuning and parameter changes, this setup does not guarantee that the generation process does not deviate from the actual scene.

As an alternative approach we tested InstructIR, which was trained to perform visual image enhancement and does not rely on a diffusion process. The problem formulation is identical to ControlNet, so InstructIR also receives an identical textual prompt for all images, and the *original prediction* to operate on. We tested both the pre-trained checkpoint of InstructIR and a version that we fine-tuned on RealEstate10k similar to ControlNet. InstructIR was trained to perform only specific image enhancement tasks like denoising and is not based on a diffusion process, so it follows the *original prediction* much more closely, while its improvements are limited this provides a higher level of temporal consistency.

5. Experiments

5.1. ControlNet

For ControlNet, we selected StableDiffusion-v2 as the base model due to its widespread usage in prior works. We consistently used a batch size of 32, requiring 30GB of RAM. Multiple experiments were conducted for $30,000$ steps with varying learning rates. We used the Adam optimizer and observed that learning rates between $1e-6$ and $5e-4$ produced reasonable outputs within this timeframe, whereas values outside this range led to poorer results. Initially, we employed the MSE loss function to predict noise; however, switching to the L_1 loss resulted in outputs that more closely followed the input, therefore we chose to continue with this loss function.

We chose not to use a prompt, our reasoning for this is twofold. First, the tasks the diffusion model has to perform is different for each image. For some images, the diffusion model needs to fill in black holes, in other images specific objects need to be reconstructed, and sometimes the input is already of good quality and doesn't need to be changed. Hand-engineering a tailor-fitted prompt for each image is labor-intensive. Second, ControlNet has been shown to work well without providing a prompt.

For both the easy and hard datasets, we trained the model for $360,000$ steps, which took more than 30 hours on our hardware. We noticed that the loss barely decreased after $30,000$ steps and that the variance is quite high across steps, the latter is typical for training and fine-tuning diffusion models (see Figure 3). Despite the minimal decrease in loss after $30,000$ steps, the model produced qualitatively better results in a non-monotonic manner. For both datasets our results are similar, but since the outputs from pixelSplat are already really good on the easy dataset, we only report the results on the hard dataset. For comparison Table 1 shows the results from pixelSplat on the two problems.

Model	PSNR ⇑	LPIPS ⇓	SSIM ⇑
pixelSplat (easy)	**26.09**	**0.1360**	**0.8630**
pixelSplat (hard)	19.38	0.2871	0.6901

Table 1. Comparison of the performance of pixelSplat (without our post-processing) on the easy and hard problems.

5.2. InstructIR

For InstructIR, we fine-tuned the provided checkpoints on our dataset. We compare the results quantitatively with and without fine-tuning.

The training time for InstructIR was significantly less than for ControlNet, taking only 3 hours. This reduction in time is attributed to the smaller model size and markedly different architecture, and additionally, the decision to freeze the language model used for producing embeddings. We selected a simple prompt [2] that provided the best qualitative outputs on a few handpicked examples and

[2]The prompt used is *Please make the image crisper and sharper*

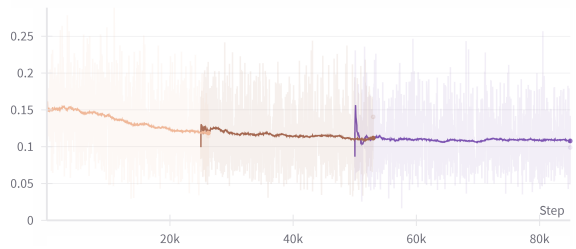

Figure 3. L_1 loss over the first 85,000 training steps of ControlNet, the different colors denote consecutive runs. The shaded area denotes the actual loss, the solid line is a running average calculated as $avg_t = 0.99 * avg_{t-1} + 0.01 * loss_t$. Note that the solid line can be misleading for the first few steps of a run.

used this prompt for all images. This approach was motivated by the uncertainty regarding the task routing mechanism for most images, leading us to rely on the model to figure out the appropriate category during training based solely on the image and constant prompt embeddings. Using a prompt as opposed to no prompt resulted in qualitatively better outputs.

We employed the L_1 loss, as in the original paper, which worked better than the MSE loss, as MSE overly focused on black pixels. The Adam optimizer with a learning rate of 10^{-5} was used for fine-tuning over 3 epochs. Frequent evaluations on the development set allowed us to select the model with the lowest loss, typically found halfway through the second epoch. Additionally, we experimented with including the LPIPS metric in the loss function with various weights but this only marginally improved the LPIPS metric on the test set while being detrimental to other metrics.

5.3. Results

5.3.1 Quantitative Results

We quantitatively compare the different methods with four metrics. Here we denote the i'th ground truth image as x_i and the corresponding predicted output image as $\hat{x}_i$. Furthermore, there are N total images with spatial resolution $R \times C$ and $x_i[r, c]$ denotes the pixel at row r and column c with RGB values normalized to be between 0 and 1. These metrics consist of:

- The L_1 loss:
  $L_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{R*C} \sum_{r=1}^{R} \sum_{c=1}^{C} |x_i[r, c] - \hat{x}_i[r, c]|$
  A lower value indicates a smaller difference between the ground truth image and the predicted image, hence lower values are preferred.
- The Peak Signal to Noise Ratio (PSNR):
  $PSNR = \frac{1}{N} \sum_{i=1}^{N} -10 * \log_{10}(MSE(x_i, \hat{x}_i))$, where
  $MSE(x_i, \hat{x}_i) = \frac{1}{R*C} \sum_{r=1}^{R} \sum_{c=1}^{C} (x_i[r, c] - \hat{x}_i[r, c])^2$
  When the MSE is small, it is assumed that there is little noise, hence the PSNR will be large, and thus higher values are preferred.
- Learned Perceptual Image Patch Similarity (LPIPS): Computes the image similarity based on the activations of a neural network. These scores have been shown to resemble how humans perceive images. Lower values indicate that the input and output are more similar.
- Structural Similarity Index (SSIM):
  In contrast to the L_1 and MSE losses that only compare individual pixels, this metric also compares how the relationship between pixels for two images differ. A higher value indicates that the images are more similar.

Table 2. Quantitative results on the full test set. The * indicates that the InstructIR model is fine-tuned on the training set. Furthermore, ⇓ indicates that lower values are better, and ⇑ that higher values are better.

Model	L_1 ⇓	PSNR ⇑	LPIPS ⇓	SSIM ⇑
pixelSplat	0.0794	19.38	**0.2871**	0.6901
InstructIR	0.0802	19.23	0.2885	0.6810
InstructIR*	**0.0724**	**19.95**	0.3175	**0.7111**
ControlNet	0.0978	17.11	0.3912	0.6094

Table 3. Quantitative results on generated views in the test set excluding black pixels. The * indicates that the InstructIR model is fine-tuned on the training set. Furthermore, ⇓ indicates that lower values are better, and ⇑ that higher values are better.

Model	L_1 ⇓	PSNR ⇑	LPIPS ⇓	SSIM ⇑
pixelSplat	0.0829	19.56	**0.3023**	0.6986
InstructIR	0.0837	19.42	0.3039	0.6888
InstructIR*	**0.0722**	**20.49**	0.3188	**0.7263**
ControlNet	0.1008	17.62	0.4078	0.6202

Table 2 showcases the outcomes of various models applied to the entire test dataset: pixelSplat, the original InstructIR model, the InstructIR model after fine-tuning on the training data, and results from ControlNet after fine-tuning on the training data. These results are interesting on many accounts.

Firstly, the original InstructIR model, designed as a versatile image enhancer, exhibits slightly inferior performance compared to the original pixelSplat results across all evaluation metrics. This is somewhat understandable, as the problem here is sufficiently different from the specific image degradations InstructIR was originally trained to fix. However, after fine-tuning on our training dataset, notable improvements can be observed. The refined outputs surpass pixelSplat results in the L_1 loss, $PSNR$, and $SSIM$, but not in LPIPS. The authors of InstructIR [8] highlighted the limitation that their model does not improve perceptual

quality, but they did not qualify this using any metrics - this is exactly what LPIPS measures. The authors also mentioned that due to the task routing architecture of InstructIR, it struggles to process images with more than one degradation correctly. These results suggest that even though InstructIR was a good baseline for quick experiments, a full training of the underlying NAFNet specifically for our use case could provide better results in the future.

Secondly, despite ControlNet taking more than 30 times longer to train, its outputs fall short compared to the fine-tuned InstructIR model. This discrepancy primarily stems from the diffusion model's excessive creativity. The diffusion model frequently produces images that are visibly different in color or brightness, which alone can result in large deviations in per-pixel metrics. It also sometimes produces entirely different objects in parts of the image, again leading to high errors.

As discussed in Section 4.2, due to the wider baseline used in our problem formulation, some of the pixelSplat output images contained large patches of black areas. Upon inspection of the image enhancement model outputs, especially for InstructIR, we noticed that it mostly fills these areas with a generic background color, without adding any details. Filling in these areas with a color closer to the original color can yield a noticeable improvement in the per-pixel results, yet this is insufficient for our needs and is not what we want to measure. To ensure that improvements in InstructIR are not purely due to this effect, we also calculated all metrics by excluding the originally black pixels. We can see in Table 3) that similar patterns emerge. This suggests that the fine-tuned InstructIR model not only addresses generic colorization in black areas but also substantially enhances the overall quality of pixelSplat's outputs.

5.3.2 Qualitative Results

In Figure 4, qualitative results from the different models are visualized. The input views for pixelSplat are displayed at the top. From these images and the information from the SfM procedure, we let pixelSplat generate 6 novel views in between these context images. These generated views are blurry as the two input views were far apart. The outputs from the fine-tuned InstructIR model in the third row are more pleasant to look at as they remove noisy parts from the reconstructed views. For example, in generated views (columns) 2, 3, and 4, there are vertical artifacts in the bottom left of the image, which the InstructIR model removes. However, the InstructIR model also makes certain correct details disappear, such as the small opening at the left side of the door in the third generated image. As discussed in both Section 5.3.1 and the original InstructIR publication [8], this is caused by the task-routing architecture of the model, which makes it struggle when having to restore multiple different kinds of degradations. While InstructIR theoretically could apply all kinds of enhancements, its training encourages the model to apply the same action globally, and we show that fine-tuning does not remove this behavior.

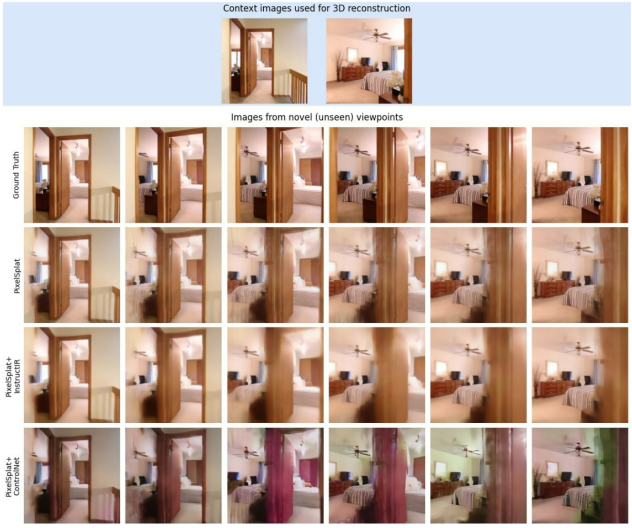

Figure 4. Samples generated with each approach, showing multiple novel views generated from the same scene. The two images in the top row (light blue background) are the *context images* used by pixelSplat. The second row shows the ground truth *target images* from 6 novel viewpoints. The third row shows the reconstructed image generated by pixelSplat, showing blur and other distortions. The last two rows show the enhanced images from the fine-tuned InstructIR model and ControlNet respectively.

The bottom row in Figure 4 shows the outputs from ControlNet for this scene. These images show that the ControlNet conditioning is strong enough for the structural similarity, however, the colors are markedly different. Additionally, due to the strong conditioning used in ControlNet, it can be seen that it re-creates and even enhances some unwanted artifacts from pixelSplat's reconstruction, like the vertical dark lines in the bottom left area of view 3.

The issues with ControlNet can be summarized into two main points. First, it fails to maintain color information. Second, it does not achieve the same level of image enhancement as InstructIR, especially when trained to replicate most of the input image. We have identified potential reasons for these shortcomings.

ControlNet introduced 'zero-convolutions' to condition the diffusion model [41], which are $1x1$ convolutions with weights and biases initialized to zero. These $1x1$ convolutions effectively preserve the input's structural information. However, because they combine channel values with learnable parameters, there is a higher likelihood of information loss in this space.

Additionally, both the control mechanism of ControlNet and the diffusion model can not explicitly detect certain

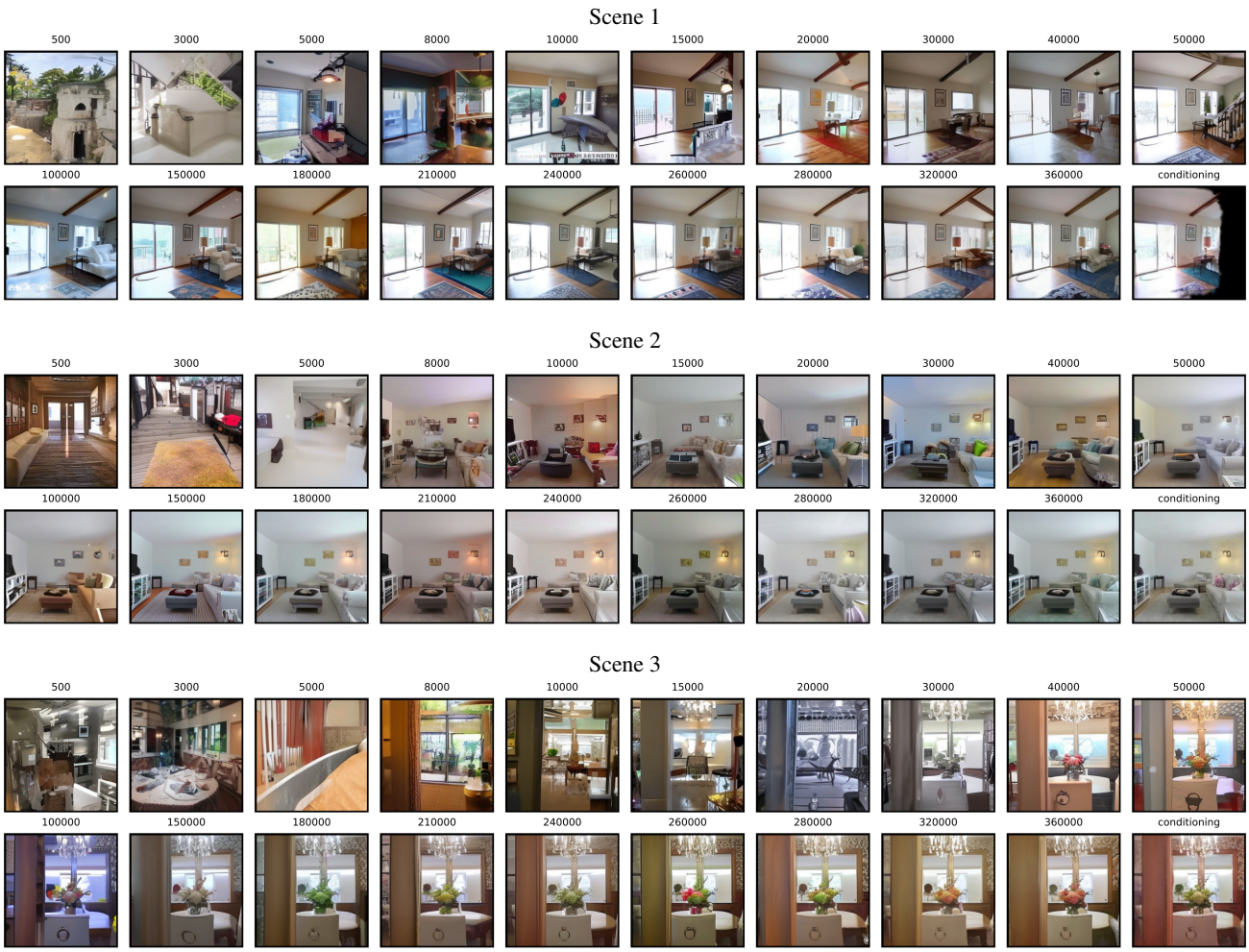

Figure 5. Output from ControlNet after a different number of update steps. The image at the bottom right is the image on which the diffusion model is conditioned. These samples are handpicked, but qualitatively representative of the other generated images for this scene after the same number of steps. Zoom in to see the small inconsistencies between the conditioning image and the generated images.

types of image degradation. However, $1x1$ convolutions lack awareness of the pixel's spatial context, which is crucial for detecting such degradations. While the diffusion model theoretically should identify real-world inconsistencies, it appears to overly rely on ControlNet's conditioning.

To address these issues, future research should focus on maintaining color information when using ControlNet conditioning and employing spatially aware conditioning. This would enable the diffusion model to better discern and utilize inputs from ControlNet, thereby more effectively filtering out real-world inconsistencies.

The 'sudden convergence phenomenon' is what the ControlNet authors dubbed the sudden learning of conditioning often observed after $15,000$ update steps. In Figure 5, we observe a similar pattern where the model begins to condition the diffusion model effectively after 15,000 steps. However, due to the greater complexity of our task compared to the creative generation of images with mild conditioning requirements, we decided to extend the training duration significantly.

Clear improvements were observed beyond this point. With $100,000$ steps, the table with the lamp in Scene 1 showed the correct shape for the first time. After an additional $50,000$ steps, the colors became more accurate, resembling the original more closely. At this stage, we believe the model had converged, as the loss, which had been decreasing only slightly before, ceased to decline further, and the results did not show qualitative improvements either.

Although the possibility of a second sudden convergence exists, we did not continue to train further due to the previously discussed limitations of ControlNet and our restricted computational resources.

5.3.3 Evaluating ControlNets' color consistency

To better understand where the conditioning of ControlNet fails, we tested the final model on single objects that were logically absent from the RE10K dataset. These images are much simpler than the ones in the dataset, making it easier to see where the model fails. We fine-tuned the test-time hyperparameters for each image to qualitatively best match the input image.

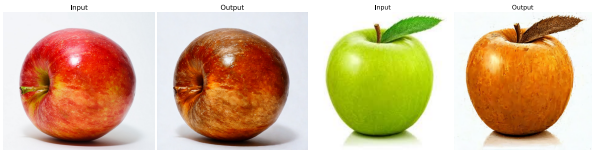

Figure 6. The input (left) and the output of ControlNet (right) with different colors of apples as input.

Figures 6, 7, and 8 all show a common trend: the structure of the input image is preserved but some colors are heavily adjusted. For example, in Figure 6 the red apple becomes a little darker and the green apple gets a brownish color. Additionally, the white background is changed to a darker tint. A possible explanation for this is that the bright red, bright green, and pure white colors were only rarely observed in the RE10K dataset and are therefore changed to colors that are typically more often found in typical home environments.

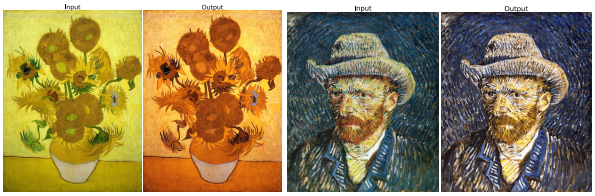

Figure 7. The input (left) and the output of ControlNet (right) with paintings as input.

Figure 7 interestingly shows that even the smallest details can be maintained by ControlNet. For both outputs, we increased the spatial resolution by a factor of two compared to the input image, resulting in crispier images. However, also here the bright colors are changed into darker ones.

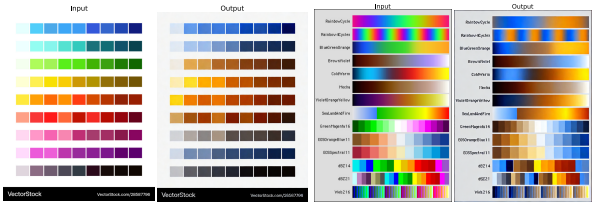

Figure 8. The input (left) and the output of ControlNet (right) with many colors as input. Zoom in to better see the results.

Finally, Figure 8 demonstrates how different colors are handled. Bright colors are altered to darker tones, resulting in the absence of pink, red, and green in the outputs. This effect is even more pronounced in the right image, where the input colors change gradually. These findings support our hypothesis that colors infrequently observed in the dataset are less likely to appear in ControlNet's output. Additional images tested on ControlNet, which were absent from the RE10K dataset, are shown in the Supplementary Material.

5.3.4 Video results

While generating videos was not an explicit aim of our project initially, we found this to be the most effective way of testing different approaches and performing qualitative analysis. We generated the novel view for each camera position of the frames of the original videos using our methods, replicating snippets of the original videos in RealEstate10k. Some of these results are available on GitHub[3]. The video results for ControlNet show strong unwanted flickering effects, as both the coloring and the structure of successive frames are significantly different. This strong flickering was what led us to also test InstructIR, which produces much smoother videos at the expense of performing much less image enhancement.

Based on the generated videos, we concluded that performing per-frame image enhancement without additional constraints that enforce temporal consistency is not sufficient for this use case. We also concluded that in future work, explicit metrics should be used to evaluate the temporal consistency of frames from the beginning of the project, to quickly identify solutions that generate plausible-looking images, but not videos.

6. Conclusion

We evaluated full-scene novel view reconstruction using pixelSplat, making the task more difficult by using context and target camera positions that are further apart from each other. The performance of pixelSplat degraded significantly in this setting compared to the original publication, which is why we explored ways to improve the reconstructed images. We implemented multiple 2D image enhancement methods, applying them as a per-frame post-processing step after generating novel views using pixelSplat. We evaluated a diffusion model based on StableDiffusion and ControlNet, which yielded visually appealing images. On the other hand, even after fine-tuning on our dataset, its output sometimes deviated significantly from the original scene, resulting in varying colors, unexpected objects, and a lack of consistency between successive frames. These artifacts

[3] https://github.com/WouterBant/diffusion-augmented-pixelsplat

ultimately led to a decrease in qualitative metrics and an unpleasant flickering in videos generated using this method.

As a potential remedy, we also tested a solution using the InstructIR model for image enhancement, which is based on a different architecture and training setup. This approach resulted in an output that was consistent with the original scene, and we achieved a small increase in three out of four metrics tracked after fine-tuning. However, it performed much smaller changes to the images and failed to correct all the mistakes we observed. Ultimately, neither method could achieve the level of improvement we aimed for.

We have identified the factors that contribute to the weaknesses of these approaches and have drawn several conclusions that can aid in solving them. Because we did not have enough time to investigate further approaches as part of the current project, we present the additional Section 7, where we discuss methods that have the potential to lead to a solution for quality enhanced consistent videos.

7. Future Work

To address the excessive creativity and inconsistency of diffusion models, we can draw inspiration from "Controllable Text-to-Video Generation with Diffusion Models" [6], which trains the model to predict entire video sequences and proposes a first-frame latent conditioning method for generating videos based on the content of the first frame. Another approach could involve adopting motion information for content consistency, as seen in "A Motion-Guided Video-to-Video Translation Framework by Using Diffusion Model with ControlNet" [20], where diffusion-model-based inpainting is used to cover residual information.

For addressing temporal consistency issues, we can look at the "Temporal-Consistent Diffusion Model for Real-World Video Super-Resolution" [42], which enhances temporal coherence by proposing a novel local-global temporal strategy within the latent diffusion framework. This model incorporates both local and global modules to preserve temporal consistency within and across video segments. Additionally, "Video ControlNet" [7] enforces temporal consistency among corresponding pixels across frames through joint noise optimization, minimizing spatial and temporal discrepancies using ControlNet and optical flow information. A new approach to addressing temporal inconsistency involves integrating autoregressiveness directly into the diffusion process, introducing autoregressive-structured correlations for the noises across time instead of using standard Gaussian noise during both training and inference stages.

Video diffusion models can also be utilized to address the temporal issue. "Imagen Video" [19] is a text-conditional video generation system that increases temporal resolution by filling in intermediate frames between input frames with its temporal super-resolution (TSR) model. This model performs temporal upsampling before concatenation by repeating frames or filling in blank frames. We can also consider the image-to-video and inpainting functionality from "Video Poet" [24], a large language model for zero-shot video generation, where the MAGVIT-v2 tokenizer enforces causal temporal dependency by encoding frames without any information from future frames. Lastly, "Stable Video 3D (SV3D)" [37] generates orbital videos around a 3D object from a single-view image, finetuning SVD [1] to create an orbital video conditioned on the initial image.

These future directions hold promise for achieving high-quality, temporally consistent video generation using diffusion models.

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023. 10

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

[3] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. *arXiv preprint arXiv:2312.12337*, 2023. 2, 3

[4] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM transactions on graphics (TOG)*, 32(3):1–12, 2013. 2

[5] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European conference on computer vision*, pages 17–33. Springer, 2022. 3

[6] Weifeng Chen, Jie Wu, Pan Xie, Hefeng Wu, Jiashi Li, Xin Xia, Xuefeng Xiao, and Liang Lin. Control-a-video: Controllable text-to-video generation with diffusion models, 2023. 10

[7] Ernie Chu, Shuohao Lin, and Jun-Cheng Chen. Video controlnet: Towards temporally consistent synthetic-to-real video translation using conditional image diffusion models. *ArXiv*, abs/2305.19193, 2023. 10

[8] Marcos V Conde, Gregor Geigle, and Radu Timofte. High-quality image restoration following human instructions. *arXiv preprint arXiv:2401.16468*, 2024. 3, 6, 7

[9] Xin Deng and Pier Luigi Dragotti. Deep convolutional neural network for multi-modal image restoration and fusion. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3333–3348, 2020. 3

[10] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4970–4980, 2023. 2

[11] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. In *Computer graphics forum*, pages 409–418. Wiley Online Library, 2008. 2

[12] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004. 1

[13] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5515–5524, 2016. 2

[14] Yasutaka Furukawa. *High-fidelity image-based modeling*. University of Illinois at Urbana-Champaign, 2008. 1

[15] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 2

[16] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 453–464. 2023. 2

[17] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 7779–7788, 2020. 3

[18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 2

[19] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022. 10

[20] Zhihao Hu and Dong Xu. Videocontrolnet: A motion-guided video-to-video translation framework by using diffusion model with controlnet, 2023. 10

[21] Chi Jin, Yuchen Zhang, Sivaraman Balakrishnan, Martin J Wainwright, and Michael I Jordan. Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences. *Advances in neural information processing systems*, 29, 2016. 3

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 3

[23] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part III 7*, pages 82–96. Springer, 2002. 1

[24] Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, Krishna Somandepalli, Hassan Akbari, Yair Alon, Yong Cheng, Josh Dillon, Agrim Gupta, Meera Hahn, Anja Hauth, David Hendon, Alonso Martinez, David Minnen, Mikhail Sirotenko, Kihyuk Sohn, Xuan Yang, Hartwig Adam, Ming-Hsuan Yang, Irfan Essa, Huisheng Wang, David A. Ross, Bryan Seybold, and Lu Jiang. Videopoet: A large language model for zero-shot video generation, 2024. 10

[25] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, pages 29–43. Wiley Online Library, 2021. 2

[26] Marc Levoy and Pat Hanrahan. Light field rendering. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 441–452. 2023. 2

[27] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021. 3

[28] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 1

[29] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[30] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 2

[31] J-P Pons, Renaud Keriven, and Olivier Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 822–827. IEEE, 2005. 1

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 3

[33] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 2

[34] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34: 19313–19325, 2021. 2

[35] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. 2

[36] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979. 1

[37] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion, 2024. 10

[38] Bin Xia, Yulun Zhang, Shiyin Wang, Yitong Wang, Xinglong Wu, Yapeng Tian, Wenming Yang, and Luc Van Gool. Diffir: Efficient diffusion model for image restoration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13095–13105, 2023. 3

[39] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[40] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022. 3

[41] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 3, 7

[42] Shangchen Zhou, Peiqing Yang, Jianyi Wang, Yihang Luo, and Chen Change Loy. Upscale-a-video: Temporal-consistent diffusion model for real-world video super-resolution, 2023. 10

[43] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 286–301. Springer, 2016. 2

[44] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 4

Improving novel view synthesis of 3D Gaussian splats using 2D image enhancement methods

Supplementary Material

8. More color consistency experiments

Figure 9. The input (left) and the output of ControlNet (right) with a tiger as input.

Figure 10. The input (left) and the output of ControlNet (right) with a tulip field as input.

Figure 11. The input (left) and the output of ControlNet (right) with a car as input.

Figure 12. The input (left) and the output of ControlNet (right) with animals as input.

Figure 13. The input (left) and the output of ControlNet (right) with an add as input.

Figure 14. The input (left) and the output of ControlNet (right) with an add as input.

Figure 15. The input (left) and the output of ControlNet (right) with sky as input.

Figure 16. The input (left) and the output of ControlNet (right) with a forest as input.