Fillerbuster: Multi-View Scene Completion for Casual Captures

Ethan Weber[1,2] Norman Müller[1] Yash Kant[1,3] Vasu Agrawal[1]

Michael Zollhöfer[1] Angjoo Kanazawa[2] Christian Richardt[1]

[1]Meta Reality Labs [2]UC Berkeley [3]University of Toronto

Incomplete Casual Capture | Our Completed Capture with Generated Novel Views

Figure 1. **Completing casual captures.** Fillerbuster takes an incomplete casual capture which has many images (left) and conditions on these to create many consistent novel views, shown on the right with arrows. The original images and the new ones enable novel-view synthesis (right) that is much more complete compared to vanilla Gaussian Splatting trained on only the incomplete casual capture (left).

Abstract

We present Fillerbuster[1], a method that completes unknown regions of a 3D scene by utilizing a novel large-scale multi-view latent diffusion transformer. Casual captures are often sparse and miss surrounding content behind objects or above the scene. Existing methods are not suitable for handling this challenge as they focus on making the known pixels look good with sparse-view priors, or on creating the missing sides of objects from just one or two photos. In reality, we often have hundreds of input frames and want to complete areas that are missing and unobserved from the input frames. Additionally, the images often do not have known camera parameters. Our solution is to train a generative model that can consume a large context of input frames while generating unknown target views and recovering image poses when desired. We show results where we complete partial captures on two existing datasets. We also present an uncalibrated scene completion task where our unified model predicts both poses and creates new content. Our model is the first to predict many images and poses together for scene completion.

1. Introduction

Photogrammetry has been around for decades [47] but only recently has become mainstream with novel-view synthesis techniques becoming high fidelity, such as NeRF [31] and Gaussian Splatting [21]. Widely used apps like Polycam [36] or Flythroughs [30] mean that everyday people can go out and easily capture content. Many such captures are done casually, which means the data is collected rather quickly and may miss large portions of the scene where the camera never looked. Sometimes, the capture is just a handful of sparse photos, which makes obtaining camera poses challenging.

Reconstructing casually captured scenes is challenging because there is missing content to complete and it is not predictable where the missing content will be from capture to capture. In contrast, the object-centric setting is much simpler as one can assume a canonical coordinate frame and sample missing views looking inward on a sphere. Instead, we highlight the challenges of scenes and focus on this more general setting, where the input camera poses can be incredibly diverse. Our goal is to fill in the missing information to enable an immersive view of the scene that feels complete, and where the rendered content can go beyond what is seen in training images, as illustrated in Figure 1. To address this

---

problem setting, we propose Fillerbuster for recovering unknown 3D information from casually captured content. This content may be casual videos, where a user quickly scans their phone through a scene, or this may be a sparse set of photos with unknown poses, e.g. from a vacation. Given this data capture as input, our unified model can jointly complete the unobserved content and recover poses.

To improve casual captures, our key insight is to jointly model the image and camera distribution of existing casual captures by using a multi-view aware diffusion model. Our approach is made possible by the large influx of captured data being recorded and uploaded online.

We design our model to handle a large and variable number of input and output frames. This is in contrast to existing generative novel-view-synthesis (NVS) methods that are typically autoregressive, meaning the next generations are conditioned on previous generations. More specifically, our problem setting is very different from the common settings of (1) generation from text only (no images) [18, 45, 58] (2) from just one image [28, 42], or (3) from two images with the goal of interpolating between them [19, 67]. We present an overview of related work in Figure 2. Our model can take in many images and camera poses, e.g. 50 images, and the user can specify which content is known, which is unknown, and which should be completed. The model can also take in partially complete images, unlike the current paradigm of assuming the input images are fully intact and known [11, 28]. Concretely, we train a large-scale diffusion transformer with a flow-matching loss for inpainting in latent space [39], conditioned on known images and camera poses (represented as raymaps) with the task of recovering the missing content, as illustrated in Figure 3.

We demonstrate our problem setting and the usefulness of our Fillerbuster model on multiple tasks. First, we show our model can complete casual captures by hallucinating large unknown regions. Second, we introduce the task of "uncalibrated scene completion", where the goal is to recover both the image poses and completed novel views. Notably, we perform both tasks with our unified model. Third, we show the multi-view inpainting task on the NeRFiller dataset [61], where we surpass prior work in quality and consistency. Finally, we present an ablation of our modeling decisions and show our model's ability to gracefully handle different numbers of input images.

2. Related work

Few-view reconstruction. Our problem setting is not few-view, but we highlight the differences here. These methods focus on deleting reconstruction artifacts [12, 40, 60], using sparse-view losses [34], leveraging depth and normal priors [53], or using generative models [20, 29, 62], to complete sparse captures. In contrast, our goal is to complete more realistic casual captures, with more input images, and to look
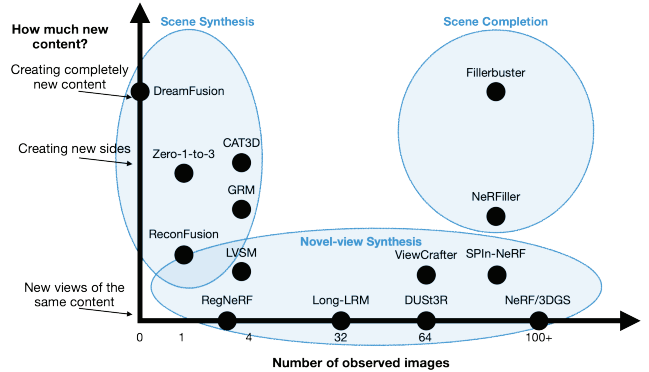

Figure 2. **Problem setting.** We illustrate our problem setting with respect to a non-exhaustive set of related work. Many works focus on scene synthesis (left) where one generates data from text or from a single image. Similarly many tackle novel-view-synthesis (bottom) to synthesize new views of the input image content. Fewer works focus on scene completion where the task is to complete missing content in captures (top right).

outside of the captured training images.

Multi-view generative models. Many single-view to single new-view models exist where an input image is known and a target image is unknown [28, 41, 44, 51]. A few methods have increased the input context to multiple input images but still generate just one output view [19, 62]. Even fewer methods have increased both the number of inputs and the number of outputs. CAT3D [11] uses 1 or 3 input images and generates 7 or 5 images, but never goes beyond a total sequence size of 8. It also remains closed-source, which limits its impact. We emphasize the importance of large sequence sizes in order to fit the entire casual capture in the context to make new content consistent with the observations. Some video models have been fine-tuned for camera control [16, 54, 59] or use geometry conditioning [27, 33, 50, 67], but these models generate smooth temporal videos, so can neither condition on the entire capture nor generate the many well-distributed views typical for the 3D reconstruction setting.

LRMs conditioned on cameras. Large reconstruction models (LRMs) that predict 3D have become popular to directly predict Gaussians [63, 69]. Most methods assume camera poses as input, which may come from traditional methods like COLMAP [43] or newer data-driven methods [24, 57]. LRMs are excellent at predicting pixel-aligned geometry but cannot inpaint unobserved areas of the scene. Furthermore, they rely on camera poses, which may be unknown in casual captures. We present a unified model for both tasks, such that when the camera is unknown, we can perform the "uncalibrated scene completion" task of making a camera fly-through of the scene from a set of sparse unposed photos. We model our camera pose prediction inspired by other data-
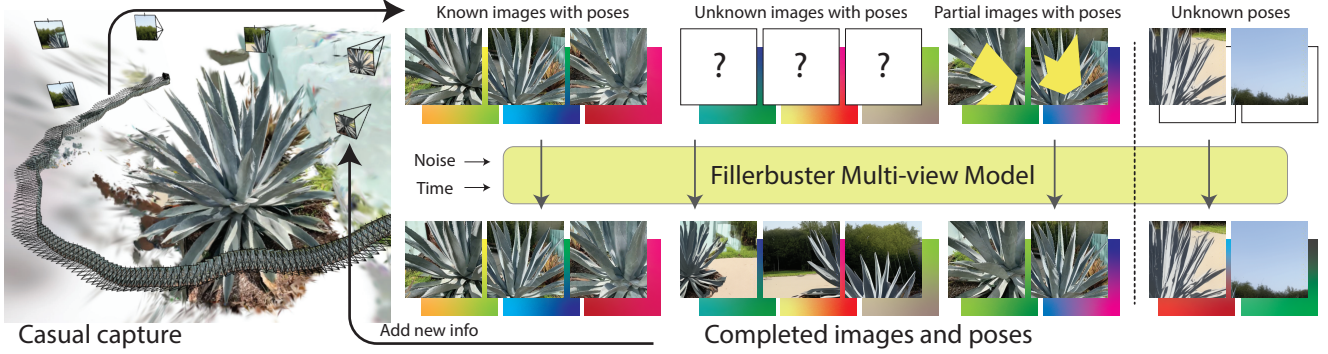
Figure 3. **Model overview.** Fillerbuster is trained on a large collection of multi-view images and poses (top and bottom of stacked images, respectively), which makes it useful for completing casual captures at inference time. More specifically, we are interested in four primary uses of the model: (1) conditioning on known images which have pose, (2) predicting new views where poses are provided, (3) predicting partial images where some pixels are known, or (4) recovering the camera poses when its unknown. Our model is a latent DiT trained to jointly model images and poses for any mixture of the input. In practice, our poses are 6-channel raymaps encoding ray origins and directions.

driven approaches [68], but use a raymap latent space, with ray origins and directions instead of Plücker coordinates.

3D inpainting. Current 3D inpainting methods such as SPIn-NeRF [32] or NeRFiller [61] rely on using 2D inpainting models within the NeRF 3D reconstruction framework to complete scenes [46]. Most methods [3–5, 25] focus on the SPIn-NeRF dataset, which is forward-facing and has much less camera motion than a typical casual capture. NeRFiller has more challenging camera movement, so we consider this dataset for experiments. However, none of these methods are conditioned on camera views when inpainting. This makes it impossible for these methods to complete scenes with large unknown content. This is because some of the generated views will be completely unknown, and without having context of the existing scene, it is unclear how to fill in the image. In contrast, our approach is camera-pose conditioned.

3. Method

We first explain our model's details for jointly modeling image completion and poses (Section 3.1), and then explain how to use Fillerbuster for casual scene completion with our model being helpful for 3D reconstruction (Section 3.2).

3.1. Fillerbuster Model

We propose a latent diffusion transformer that denoises multiple input images and calibrated camera poses with masks indicating known and unknown regions. There are N elements in a sequence with images $I_i \in \mathbb{R}^{H \times W \times 3}$, raymaps $R_i \in \mathbb{R}^{H \times W \times 6}$ with origin and direction per pixel, valid image masks $\mathcal{M}_i^I \in \mathbb{R}^{H \times W}$, and valid ray masks $\mathcal{M}_i^R \in \mathbb{R}^{H \times W}$, where 1 indicates known conditioning information and 0 indicates unknown pixels. Our goal is to predict all images and raymaps given only the known information, i.e., $p(I, R \mid I \odot \mathcal{M}^I, R \odot \mathcal{M}^R)$. We use "sequence" to refer to multiple images and cameras from the same capture, and "sequence size" for how many images are denoised together.

Model architecture. The architecture is designed for latent inpainting [39], taking in any combination of known and unknown images and raymaps, and predicting the missing values. We use a DiT architecture [35] and train with the flow matching objective [26]. We train separate VAEs for images and poses encoded as raymaps, where $\mathcal{E}^I$ denotes the image encoder and $\mathcal{E}^R$ denotes the raymap encoder. Both encoders compress the spatial resolution by a factor of $8\times$ and output a d-dimensional representation. We set $d = 16$ for both encoders. Let $z_i^I = \mathcal{E}^I(I_i)$ and $z_i^R = \mathcal{E}^R(R_i)$ denote the compressed latent image and raymap, respectively. Let $\mathcal{D}$ denote a downscaling operation that reduces the spatial resolution by the same factor as the encoders. We add noise to z_i as $\tilde{z}_{i,t} = (1 - t)z + t\epsilon$, then prepare the sequence as

$$
\begin{aligned}
s_{i,t} = \tilde{z}_{i,t}^I \oplus \mathcal{E}^I(I_i \odot \mathcal{M}_i^I) \oplus \mathcal{D}(\mathcal{M}_i^I) \oplus \\
\tilde{z}_{i,t}^R \oplus \mathcal{E}^R(R_i \odot \mathcal{M}_i^R) \oplus \mathcal{D}(\mathcal{M}_i^R),
\end{aligned}
\tag{1}
$$

where $\oplus$ denotes concatenation of the noisy latents, known image and ray latents, and the masks themselves. The noisy sequence $s_t \in \mathbb{R}^{N \times H \times W \times (4d+2)}$ is patchified, and positionally embedded (described later), then passed through the transformer model $\mathcal{F}$ to predict the denoised latent images and raymaps as $\{z^I, z^R\} = \mathcal{F}(s)$. Our VAEs have a convolutional architecture [39] and train with KL [23], adversarial [13], and L1 reconstruction losses. Our transformer architecture is "DiT-L/2" [35] with a latent patch size of 2×2 and 24 layers of multi-head self-attention. Our model only has 650M parameters – small enough to fit on most GPUs and fast enough to use with open-source 3D reconstruction tools.

Raymap coordinate convention. Raymaps comprise per-pixel ray origins and world-space unit directions. At training time, we randomly choose one camera from our sequence to be at the origin and oriented upright. We also randomly rotate and rescale the cameras for augmentation, and ensure that origins are always within the cube $[-1, 1]^3$.

Figure 4. **Model samples.** Here we show generations from our model. For this setting, we provide pose input for all images. The top rows indicates which pixels are known, with yellow indicating unknown regions. The middle rows show the inpainted images after passing the entire sequence of size 16 (top rows) into the model for 24 denoising steps. The bottom rows show the ground truth, but note that this is not necessarily the only correct solution if the newly generated pixels are unobserved according to the masks. Notice that in the top example, the generations are self-consistent but different than the GT, which is entirely plausible.

Masking out regions. During training, we mask out information from the images and/or the raymaps; the task is to predict the denoised sequence from partial noise. We apply masking at the pixel level before VAE encoding (Equation (1)) to enable precise control over which pixels are known or unknown. We use a mixed masking strategy: some images are known, some are unknown, and some are partially unknown with randomly rotated rectangles, as illustrated in Figure 4. We dropout image and raymap masking with a 10% chance to enable classifier-free guidance [17].

Token positional embeddings. We use two forms of positional embeddings to enable varying sequence lengths to be generated at inference time. *1) 2D layout embeddings* encode the layout of the image with fixed sinusoidal embeddings. *2) Index embeddings* are more unique for our setting, where we add an unordered index descriptor to each token coming from the same image. More specifically, the full sequence s is first patchified and projected into patches p. It is then augmented with positional embeddings as $p' = \psi_{2D}(p) + \psi_{Idx}(p)$, where ψ_{2D} is sinusoidal embeddings to encode the 2D layout of each patch within the image itself [55], and ψ_{Idx} to encode which index in the sequence the patch is from. During training, $\psi_{Idx}(p)$ randomly samples a frequency for each image and then adds that value to each patch of the same image. During inference, the frequencies are chosen with uniform spacing and applied in the same way so each image has a unique identifier. This helps support generating longer sequences at inference time beyond the training lengths, which we show in Section 4.4. Prior multi-view diffusion transformer models do not incorporate this, and we show that this is useful for generating longer sequences.

Training and inference details. We train our model *from scratch* on a collection of datasets including ScanNet++ [65] and a corpus of Shutterstock data including 2D images and 3D asset renderings. We train our final model on 64 A100 GPUs for approximately a month. We first train at $256{\times}256$ resolution for 1M iterations, and then fine-tune for 100K iterations with resolutions varying from $64{\times}64$ to $1024{\times}1024$ with sequence lengths between 20 and 2, depending on how many images fit in GPU memory for a given resolution. See our appendix for additional details. For inference, we apply classifier-free guidance (CFG) by dropping out both image and raymap conditioning for an unconditional prediction. We use spatially varying CFG weights of 7 for the unknown regions and 1.1 for the known regions to avoid saturation artifacts since the task of copying the conditioning is much easier than predicting new information [2].

3.2. Multi-View Scene Completion

Here we explain how to use our model to complete scenes. **Variable sequence lengths.** Our index embedding enables changing the sequence length at inference time. We leverage this property to generate many images at the same time for inpainting incomplete scenes, since NeRF and Gaussian splatting typically require many views to create a scene. **Multi-view inpainting for scene completion.** We complete scenes by generating novel views and adding them to our existing dataset, then optimizing 3DGS [21]. We avoid the need for an SDS-like optimization approach [14, 37] because our model can generate many consistent images with a large sequence size. To complete scenes with large camera movement, we add new views to the scene that look in all di-
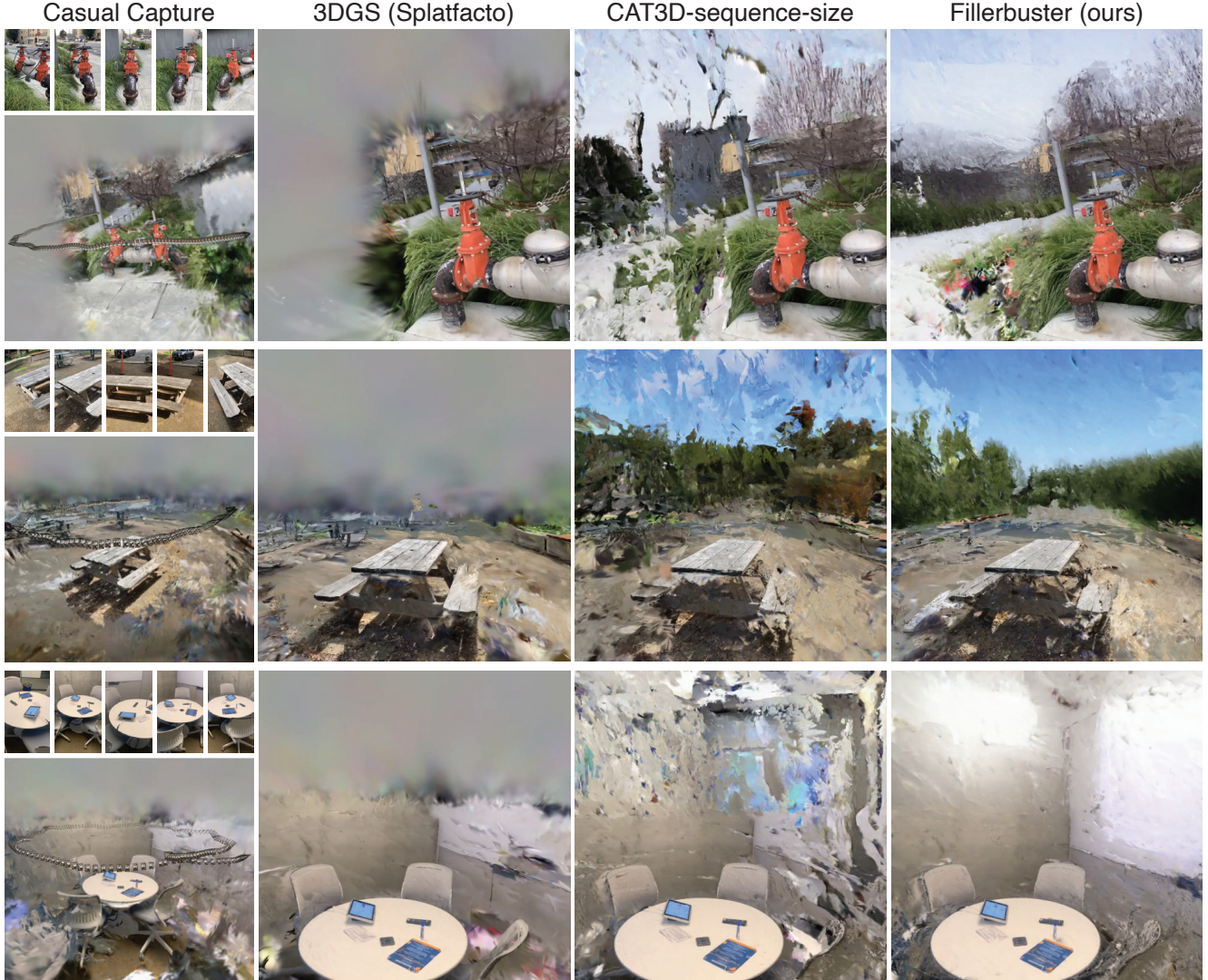
Figure 5. **Completing casual captures.** Here we demonstrate our ability to complete casual captures from the training splits of the Nerfbusters dataset [60]. On the left, we show the input captures and some representative images. 3DGS (Splatfacto) cannot add missing details so the capture remains incomplete. Our CAT3D baseline conditions on 3 images and generates 6 images at a time, so it cannot produce consistent content. Fillerbuster conditions on 16–40 images to generate 24 novel views, and obtains the most consistent results.

rections. We first inpaint many (∼25) "anchor" frames, and then condition on these frames to generate more novel views, as in CAT3D [11]. The key difference is that we can handle much larger sequence sizes than CAT3D, which operates on at most 3 images for conditioning. Furthermore, unlike CAT3D, we can also complete scenes with partial masks. To complete these scenes, we inpaint the images themselves and update the dataset with the new pixels.

Normal regularization. We find that regularizing Gaussian splat geometry towards the end of optimization can help improve results. Specifically, we apply a total-variation smoothness loss on rendered normals [10], and we also align our depth-derived surface normals with rendered normals (similar to Verbin et al. [56] but using Gaussians instead

of NeRF). This second loss is $\mathcal{L}_{\text{align}} = \|\text{sg}(N_{\text{r}}) - N_{\text{d}}\|_2^2 + \|\text{sg}(N_{\text{r}}) - N_{\text{d}}\|_2^2$, where N_{r} are rendered normals from 3D Gaussians, oriented towards the camera, and N_{d} are normals derived from rendered depth maps. We apply our normal regularizations after the initial geometry has taken form, at approximately 10K steps. Please see the appendix for more details.

4. Evaluation

We first show our casual scene capture completion results on the Nerfbusters dataset, and then demonstrate the "uncalibrated scene completion" task on data captured ourselves. Next, we show results on the NeRFiller dataset, where we surpass prior work in quality and consistency. Finally, we evaluate our model design choices. Note that we choose to
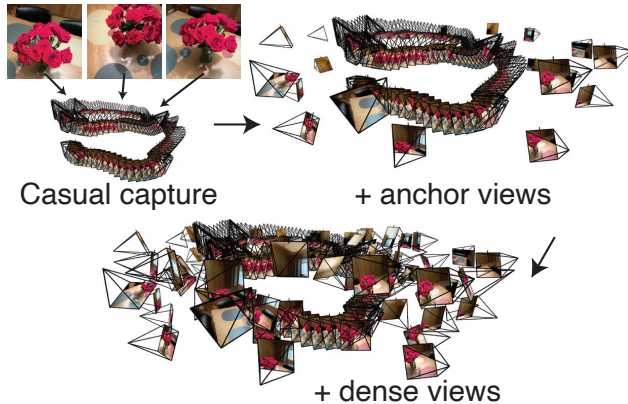
Figure 6. **Novel-view sampling.** We start with a casual capture (top left) and condition on 16 of the images to generate 24 anchor views simultaneously (top right). We then condition on the casual capture and anchors to densify views (bottom). We repeat the dense stage for multiple rounds to reach ~100 novel views in total.

use 3D Gaussian splatting [21] for our reconstruction experiments rather than NeRF [31] because 3DGS is fast to train and thus gaining popularity among casual capture users.

4.1. Completing Casually Captured Scenes

Setting. Here we show results for completing casually captured scenes. We choose the Nerfbusters dataset [60] for this setting because it mimics the casual captures of an inexperienced user. Our goal is to take these partial captures and to complete them – either by completing geometry or adding context to the capture. We compare the following methods: (1) *3DGS* (Splatfacto [49], which uses the gsplat library [64], with no inpainting), (2) *NeRFiller* [61] (NeRFiller inpainting, which we note is not suitable for this setting where the new views do not have partial masks), (3) *CAT3D-sequence-size* (ours with CAT3D-sized conditioning, where we condition on 3 images and generate 6 images a time, further described in the appendix), and (4) *Fillerbuster* (our complete method, where we condition on 16 views and generate 24 images at a time). We perform multiple rounds of inpainting to reach ~100 new views that are added to the scene. We show this procedure in Figure 6, where we sample cameras on a cylinder looking at random directions. Unfortunately, CAT3D [11] is not open-sourced, so we cannot compare with it directly.

Results. We show qualitative results in Figure 5. We find that *3DGS* cannot add any additional detail, leading to large unknown regions when rendering novel views away from the training images. Naïvely adapting *NeRFiller* to this challenging setting fails drastically because the inpainting is not conditioned on pose and are therefore random, adding random colors to the scene. *CAT3D-sequence-size* is more consistent but introduces artifacts due to the limited context size. Our proposed method *Fillerbuster*, with large sequence sizes for conditioning and generation, of the most consis-
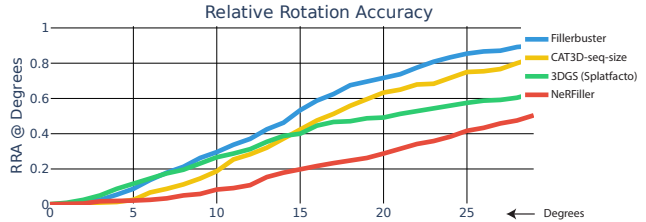

Figure 7. **Completing casual captures metrics.** We report relative rotation accuracy for nearby frames in a novel-view video. We use off-the-shelf correspondences [48] to estimate camera rotation and compare with the ground truth. Fillerbuster produces the most consistent videos from a pose-estimation perspective.

tent. We design a new metric, to evaluate this task since we do not have ground-truth when hallucinating novel scene content. We are inspired by previous work that measures reconstruction error [9] or distance to epipolar lines [33, 66]. Specifically, we render a novel-view camera trajectory and estimate the poses between nearby frames. We use off-the-shelf correspondences [48] and classical methods [15] to obtain a relative rotation, which we compare with the ground truth. We report relative rotation accuracy in Figure 7 and find the qualitative results to be consistent with camera-pose estimation accuracy.

4.2. Uncalibrated Scene Completion

Here we consider the task of scene completion, starting from a collection of 16 unposed photos. We show that our unified image-and-pose model supports such casual captures by predicting camera poses and then generating a fly-through of the scene, completing unknown content where it's missing.

Setting. Given the set of images, we can denoise the raymaps conditioned only on the images. We use joint denoising tiling with a window size of 8 images and average 8 times per denoising step. This is similar to MultiDiffusion [1] or NeRFiller's joint denoising [61] (see appendix). Then, we solve for the pinhole camera parameters that match backprojected rays to the denoised rays, taking only 5 seconds to converge for 16 images. Next, we condition on our predicted rays to generate novel views to complete the scenes. We create a camera path by fitting a 2D ellipse to our posed images and point cameras inward. Here we use a sequence size of 48: 16 input images with generated poses, plus 32 generated images with specified poses, but note that this decision is flexible.

Results. Our joint modeling of poses and images is convenient because we do not rely on external structure-from-motion; instead, our unified model can handle both tasks gracefully. We show qualitative results of our video poses in Figure 8. Please see our project page for video results.

4.3. Completing Masked 3D Regions

Figure 9 shows results where we inpaint scenes from the NeRFiller dataset and compare against the NeRFiller method
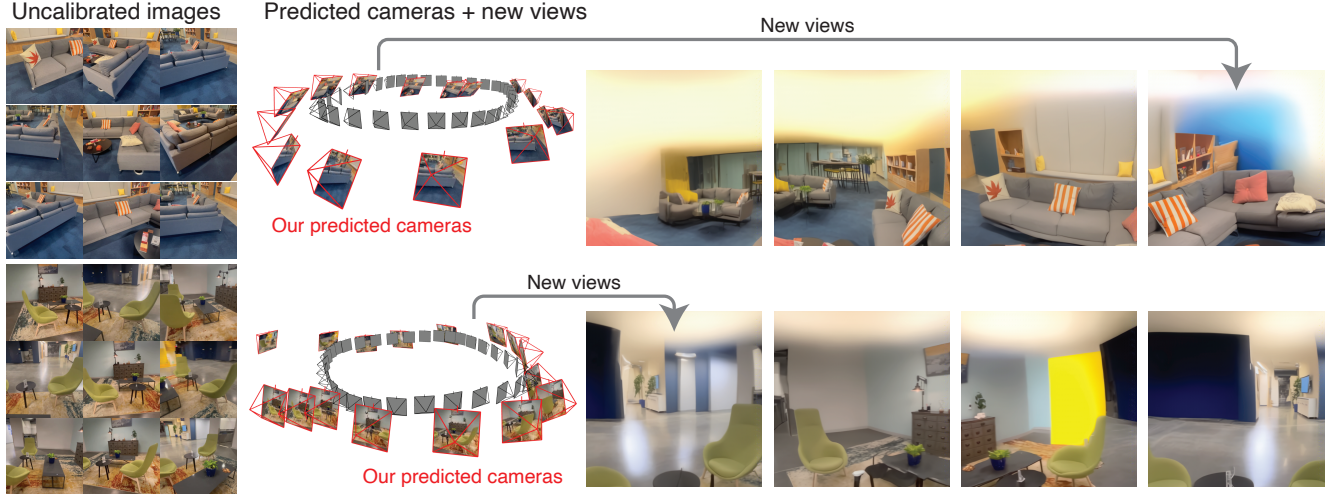
Figure 8. **Uncalibrated scene completion.** We capture some scenes with an iPhone 14 Pro and run our framework. We start from 16 uncalibrated and unposed images (left), and we use our model to both predict camera pose (middle) and generate completed views (right). We show our predicted cameras in red compared and unknown views we will sample in black. Our cameras are plausible and useful for conditioning on to generate new views. We show just 4 views here and the full videos on the project page.
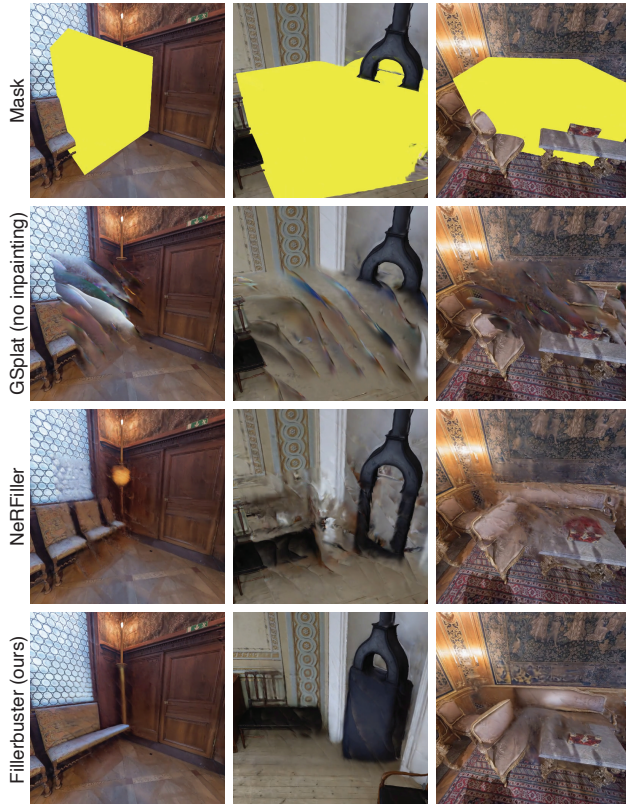

Figure 9. **NeRFiller dataset novel-views.** We complete NeRFiller scenes [61] with higher quality and control than their method.

[61]. For both NeRFiller and Fillerbuster, we inpaint 32 equally-spaced training images and then train 3D Gaussian splatting with our normal regularization. Notably, unlike in NeRFiller, we do not use depth supervision or iterative

Table 1. **Completing masked 3D regions.** On the NeRFiller dataset [61], we report novel-view synthesis metrics where we compare the rendered images with the inpainted images. In parentheses, we report numbers without using our normal regularizations. No normal regularization lets the network cheat to explain inconsistencies, leading to slightly improved but misleading metrics. Overall, we find Fillerbuster is much more consistent than NeRFiller.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRFiller	25.57 (25.94)	0.89 (0.88)	0.182 (0.194)
Fillerbuster	**29.60** (30.65)	**0.92** (0.93)	**0.096** (0.069)

dataset updates. We instead inpaint once at the start of training to directly assess multi-view inpainting quality, regardless of any SDS-style optimizations that encourage consistency. We also report reconstruction metrics in Table 1 by comparing our 32 inpainted images with the final renderings from the same 32 viewpoints. Our method is more consistent than NeRFiller, with and without our normal regularization.

4.4. Model Design Ablations

We evaluate our model choices with the Nerfstudio dataset [49] because it consists of 10 well-captured static scenes that look in all directions. In Table 2, we report novel-view synthesis metrics for 256×256 resolution images for varying sequence lengths. For each scene, we randomly sample a sequence of size N image crops of this resolution. We condition on $N/4$ full crops and $N/4$ partial crops, and generate all the missing information (see Figure 4 for examples). We repeat this procedure 50 times for sequences of length $N \in \{8, 16, 32\}$, and report averaged metrics. Each model is trained from scratch for 100K iterations with sequences of $N = 8$ images. Inspired by Esser et al. [8], we also report

Table 2. **Model design ablations.** We evaluate our model on posed images from the Nerfstudio Dataset [49]. We show a Fillerbuster prediction above the table, where we compare the generation vs. the ground truth for reconstruction metrics (PSNR/SSIM/LPIPS). For hallucination metrics, we report the conditional validation loss (VAL) as done by Esser et al. [8]. Notably, we focus on image generation rather than pose prediction but find that not predicting pose ("no-pose-pred") leads to worse results. See Section 4.4 for detailed descriptions.

| Method	Iters	8-views				16-views				32-views			
		PSNR ↑	SSIM ↑	LPIPS ↓	VAL ↓	PSNR ↑	SSIM ↑	LPIPS ↓	VAL ↓	PSNR ↑	SSIM ↑	LPIPS ↓	VAL ↓
no-index-emb	100K	11.10	0.431	0.456	0.2394	14.06	0.450	0.400	0.2417	12.38	0.467	0.422	0.2438
fixed-index-emb	100K	10.46	0.353	0.520	0.2517	12.14	0.390	0.491	0.2546	12.52	0.416	0.448	0.2556
no-poses	100K	11.63	0.413	0.426	0.2386	14.73	0.461	0.384	0.2411	13.39	0.476	0.389	0.2431
random-poses	100K	11.82	0.431	**0.415**	0.2384	**16.18**	**0.487**	0.333	0.2409	**14.27**	0.483	0.367	0.2430
Fillerbuster	100K	**11.97**	**0.435**	**0.415**	**0.2383**	15.81	0.481	**0.329**	**0.2407**	14.21	**0.486**	**0.366**	**0.2426**
Fillerbuster	1M	12.77	0.442	0.381	0.2365	17.20	0.485	0.281	0.2388	14.13	0.498	0.352	0.2396
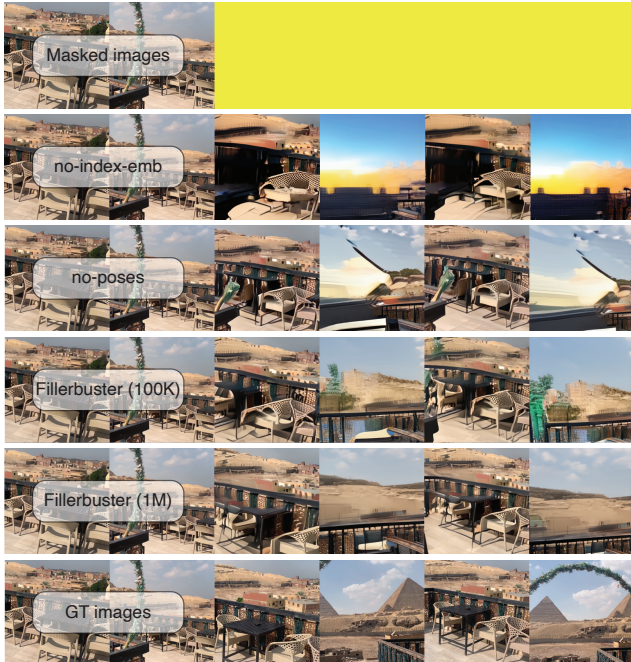
Figure 10. **Qualitative results for model ablations**. We provide pose for all images and perform completion in the unknown regions (yellow). Without index embeddings, the model fails to reason about which image the tokens are coming from, so the results are patchy and blurry. Without training for pose prediction, the generations are worse than training for pose prediction (Fillerbuster (100K)). The last rows show our final model and GT for reference.

validation losses ("VAL") on these samples for 20 equally-spaced time steps. This metric captures generation quality unlike the other metrics that evaluate reconstruction.

Table 2 compares the following ablations (see Figure 10 for visual examples): *"no-index-emb"* does not use index embeddings and instead relies on the raymaps to understand the token relationships. This makes the task harder and the model performs worse. *"fixed-index-emb"* uses a fixed number of index embeddings, preventing it from generalizing to more than 8 images. To go beyond 8 images, more index embeddings are introduced, which this model variant cannot handle. Notice the low PSNR of this setting for 16-views. *"no-poses"* does not denoise raymaps, and interestingly, we

find that when it does not learn to predict camera pose, the model performs worse at image generation, indicating that image and pose predictions are complementary tasks. Finally, *"random-poses"* randomizes the poses instead of forcing them to be upright with one camera at the origin. Our final model is trained for much longer and is shown at the bottom of the table, obtaining the best "VAL" results. We note that the metrics vary from 8-views to 32-views because more conditioning and more unknowns are introduced, and reconstruction metrics are imperfect for assessing the models.

5. Conclusion

Many 3D casual captures are missing content because the camera does not look everywhere. To recover these missing details, we present Fillerbuster, a large-scale multi-view diffusion model, to complete missing regions or recover camera poses when they are not available. We show our model is useful at completing casual captures, and we introduce an "uncalibrated scene completion" task where we generate novel missing content from unposed images from our own mobile captures. We also outperform NeRFiller on its setting, where partial views are known and there are masks to complete. Lastly, we present important model design decisions to enable large sequence conditioning and generation.

The area of scene completion is incredibly exciting and there are many avenues to explore beyond what is presented here. For example, choosing camera paths can be challenging because cameras should not be sampled inside of objects or behind walls. For the Nerfbusters dataset, we sample on a cylinder (Figure 6), but this may not be suitable for more catered settings, such as looking under a desk or behind a corner. Consequently, a method that predicts where to sample next could be valuable. We also note that our model's generations become worse when the generated cameras are very far away from the conditioning views, e.g. when we step back too far from the input views. Incorporating more diverse training data, or specifically rendering distant viewpoints in simulations, may be useful. We proposed an initial step towards the challenging problem of casual capture scene completion. We expect the results will further improve with a larger model, bigger compute budget, and more diverse training data.

References

[1] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. MultiDiffusion: Fusing diffusion paths for controlled image generation. In *ICML*, 2023. 6

[2] 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. *arXiv preprint arXiv:2311.15127*, 2023. 4

[3] Chenjie Cao, Chaohui Yu, Yanwei Fu, Fan Wang, and Xiangyang Xue. MVInpainter: Learning multi-view consistent inpainting to bridge 2D and 3D editing. In *NeurIPS*, 2024. 3

[4] Honghua Chen, Chen Change Loy, and Xingang Pan. MVIP-NeRF: Multi-view 3D inpainting on NeRF scenes via diffusion prior. In *CVPR*, 2024.

[5] Jiafu Chen, Tianyi Chu, Jiakai Sun, Wei Xing, and Lei Zhao. Single-mask inpainting for voxel-based neural radiance fields. In *ECCV*, 2024. 3

[6] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *CVPR*, pages 13142–13153, 2022. 1

[7] Matt Deitke et al. Objaverse-XL: A universe of 10M+ 3D objects. In *NeurIPS*, 2024. 1

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 7, 8, 1

[9] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. In *NeurIPS*, 2024. 6

[10] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022. 5

[11] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. CAT3D: Create anything in 3D with multi-view diffusion models. In *NeurIPS*, 2024. 2, 5, 6, 3

[12] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields. In *CVPR*, pages 20061–20070, 2024. 2

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 3

[14] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, pages 19740–19750, 2023. 4

[15] Richard I Hartley. In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19(6):580–593, 1997. 6

[16] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024. 2

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4

[18] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In *ICCV*, 2023. 2

[19] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3D inductive bias. *arXiv preprint arXiv:2410.17242*, 2024. 2

[20] Yash Kant, Aliaksandr Siarohin, Michael Vasilkovsky, Riza Alp Guler, Jian Ren, Sergey Tulyakov, and Igor Gilitschenski. iNVS: Repurposing diffusion inpainters for novel view synthesis. In *SIGGRAPH Asia*, 2023. 2

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 4, 6

[22] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. LERF: Language embedded radiance fields. In *CVPR*, pages 19729–19739, 2023. 1

[23] Diederik P Kingma. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[24] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3D with MASt3R. In *ECCV*, 2024. 2

[25] Chieh Hubert Lin, Changil Kim, Jia-Bin Huang, Qinbo Li, Chih-Yao Ma, Johannes Kopf, Ming-Hsuan Yang, and Hung-Yu Tseng. Taming latent diffusion model for neural radiance field inpainting. In *ECCV*, 2024. 3

[26] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2022. 3

[27] Fangfu Liu, Wenqiang Sun, Hanyang Wang, Yikai Wang, Haowen Sun, Junliang Ye, Jun Zhang, and Yueqi Duan. ReconX: Reconstruct any scene from sparse views with video diffusion model. *arXiv preprint arXiv:2408.16767*, 2024. 2

[28] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *ICCV*, pages 9298–9309, 2023. 2

[29] Xinhang Liu, Jiaben Chen, Shiu hong Kao, Yu-Wing Tai, and Chi-Keung Tang. Deceptive-NeRF/3DGS: diffusion-generated pseudo-observations for high-quality sparse-view reconstruction. In *ECCV*, 2024. 2

[30] Inc. Luma AI. Luma flythroughs, 2024. 1

[31] 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. 1, 6

[32] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. In *CVPR*, pages 20669–20679, 2023. 3

[33] Norman Müller, Katja Schwarz, Barbara Rössle, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. MultiDiff: Consistent novel view synthesis from a single image. In *CVPR*, pages 10258–10268, 2024. 2, 6

[34] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, pages 5480–5490, 2022. 2

[35] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pages 4195–4205, 2023. 3

[36] Polycam. LiDAR & 3D scanner for iPhone & Android, 2024. 1

[37] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023. 4

[38] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for PyTorch. In *WACV*, pages 3674–3683, 2020. 3

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2, 3, 1

[40] Sara Sabour, Lily Goli, George Kopanas, Mark Matthews, Dmitry Lagun, Leonidas Guibas, Alec Jacobson, David J Fleet, and Andrea Tagliasacchi. SpotlessSplats: Ignoring distractors in 3D Gaussian splatting. *arXiv preprint arXiv:2406.20055*, 2024. 2

[41] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, et al. ZeroNVS: Zero-shot 360-degree view synthesis from a single real image. *arXiv preprint arXiv:2310.17994*, 2023. 2

[42] 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 real image. In *CVPR*, 2024. 2

[43] Johannes L Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 2

[44] Junyoung Seo, Kazumi Fukuda, Takashi Shibuya, Takuya Narihira, Naoki Murata, Shoukang Hu, Chieh-Hsin Lai, Seungryong Kim, and Yuki Mitsufuji. GenWarp: Single image to novel views with semantic-preserving generative warping. In *NeurIPS*, 2024. 2

[45] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3D generation. In *ICLR*, 2024. 2

[46] Meng-Li Shih, Wei-Chiu Ma, Aleksander Holynski, Forrester Cole, Brian L. Curless, and Janne Kontkanen. ExtraNeRF: Visibility-aware view extrapolation of neural radiance fields with diffusion models. In *CVPR*, 2024. 3

[47] Peter Sturm. A historical survey of geometric computer vision. In *International Conference on Computer Analysis of Images and Patterns (CAIP)*, 2011. 1

[48] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, pages 8922–8931, 2021. 6, 3

[49] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David Mcallister, Justin Kerr, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023. 6, 7, 8, 1, 2, 3

[50] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. MVDiffusion: Enabling holistic multiview image generation with correspondence-aware diffusion. In *NeurIPS*, 2023. 2

[51] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Josh Tenenbaum, Frédo Durand, Bill Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. In *NeurIPS*, pages 12349–12362, 2023. 2

[52] The Movie Gen team @ Meta. Movie Gen: A cast of media foundation models. arXiv:2410.13720, 2024. 1

[53] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. DN-splatter: Depth and normal priors for Gaussian splatting and meshing. In *WACV*, 2025. 2

[54] Basile Van Hoorick, Rundi Wu, Ege Ozguroglu, Kyle Sargent, Ruoshi Liu, Pavel Tokmakov, Achal Dave, Changxi Zheng, and Carl Vondrick. Generative camera dolly: Extreme monocular dynamic novel view synthesis. In *ECCV*, 2024. 2

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[56] Dor Verbin, Pratul P Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T Barron. NeRF-Casting: Improved view-dependent appearance with consistent reflections. In *SIGGRAPH Asia*, 2024. 5

[57] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, pages 20697–20709, 2024. 2

[58] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *NeurIPS*, 2023. 2

[59] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tian-shui Chen, Menghan Xia, Ping Luo, and Ying Shan. MotionCtrl: A unified and flexible motion controller for video generation. In *SIGGRAPH*, 2024. 2

[60] Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured NeRFs. In *ICCV*, pages 18120–18130, 2023. 2, 5, 6

[61] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. Nerfiller: Completing scenes via generative 3D inpainting. In *CVPR*, pages 20731–20741, 2024. 2, 3, 6, 7

[62] 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. In *CVPR*, pages 21551–21561, 2024. 2

[63] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large Gaussian reconstruction model for efficient 3D reconstruction and generation. In *ECCV*, 2024. 2

[64] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 6, 3

[65] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, pages 12–22, 2023. 4, 1

[66] Jason J Yu, Fereshteh Forghani, Konstantinos G Derpanis, and Marcus A Brubaker. Long-term photometric consistent novel view synthesis with diffusion models. In *ICCV*, pages 7094–7104, 2023. 6

[67] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. ViewCrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024. 2

[68] Jason Y Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *ICLR*, 2024. 3

[69] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-LRM: Long-sequence large reconstruction model for wide-coverage Gaussian splats. arXiv:2410.12781, 2024. 2

A. Appendix

Here we provide more information about our paper. There is additional info at our project page, https://ethanweber.me/fillerbuster/.

B. More Results

Please see the website for interactive results where we show more results in the Nerfbusters and Nerfiller datasets. We also have a new experiment in a section called **"Flexible Conditioning and Generation"** where we illustrate how our model can adapt to different number of input frames. We show results for this on the LERF dataset [22].

C. Training Recipe

Here we provide more training details about our model. Some of the data we use are as follows:

- **ScanNet++ [65]:** ∼500 captures. This dataset is a collection of indoor scenes. We use their high-quality DSLR images data for training our model.
- **Shutterstock3D:** ∼2M 3D assets from Shutterstock. For the meshes, we render them from 24 views sampled on a sphere using Blender and the Cycles physically-based path tracer, similar to Objaverse [6, 7].
- **Shutterstock2D:** ∼400M image-text pairs from Shutterstock. These images are aesthetically pleasing. We use these images since they have text information and can help learn the long-tail of information that may not be present in our other datasets.

We combine these datasets with various other multi-view data available to us. We train our model for multi-view image and raymap denoising. We include single-view data in the training mix because we are training the model from scratch and wish to learn more concepts outside of of the more specific multi-view data. 50% of the time, the model samples multi-view data, and the other 50% of the time, the model samples single-view data.

Multi-view data sampling. We implement our multi-view dataloader by leveraging the Nerfstudio [49] framework. We use custom Nerfstudio DataParsers for each dataset type and train NeRFs on a subset of the captures with the Nerfacto [49] method. After confirming that our image crops and rays are sampled properly, we can confidently use the dataset for training our multi-view diffusion model. Within each of these multi-view datasets, we uniformly sample frames with one of 5 strides: 1, 2, 4, 8, or completely random (i.e. no stride). We sample random crops within each image, and with 10% chance we center the crop. With 10% chance, we drop out conditioning. 75% of the time, we train for image denoising, and 25% of the time, we train for raymap denoising.

Single-view data sampling. Our single-view data from Shutterstock2D is treated as a single-sequence set with a text

prompt and unknown camera ray conditioning. In this case, we mask out the raymaps as conditioning and we also mask out the loss for the noise prediction, to not penalize the predictions where we do not have ground truth. We use cross-attention with text embeddings. In practice, we only use the text to control coarse signals like like brightening the generation by using the word "bright".

Mixed-precision training. We train with `bfloat16` to reduce memory requirements and speed up training. To stabilize training, we found it was important to perform LayerNorm in `float32`, and to normalize keys and queries before attention operations [8].

Hyperparameters. We use flow matching and logit normal sampling [8]. Our learning rate is constant at 10^{-4}. For multi-view training at 256^2 resolution, we use a batch size of 4 sequences, and a sequence size of 10 for each GPU. For single-view training at 256^2 resolution, we use a batch size of 52 images, where in this case the sequence size is 1. We train with 64 A100 GPUs across 8 nodes for 1M steps which takes roughly a month. Finally, we train for an additional 100K steps with varying image resolution. To implement this, we assign each GPU to a specific resolution with uniform probability and tune the batch size and sequence size to fit within the GPU memory. We use the sequence sizes {20, 20, 10, 5, 2} for resolutions {64^2, 128^2, 256^2, 512^2, 1024^2}, respectively.

D. Method Details

D.1. Image and Raymap VAE

We train two separate VAEs from scratch for compressing images and raymaps into latent representations. We use the same convolutional architecture [39] for both VAEs, each with 4 down blocks and 4 up blocks. For the image VAE, the output channels for each down block are 128, 256, 512, and 512. We use 3 input channels and 16 dimensions for the latent dimension. For the raymap VAE, we reduce the down block channel dimensions by a factor of 4, leading to dimensions: 32, 64, 128, and 128. The raymap VAE takes as input 6 channels and has a latent dimension of 16. For the raymap VAE, we remove group norm since we noticed it produced spot artifacts in the corner, consistent with previous findings [52], and we modified the padding in the network to use "replicate" padding. The image VAE has 84M parameters and the raymap VAE has 5M parameters. The image VAE trains on all our data, while the raymap VAE trains on only our multi-view data.

D.2. Model Architecture

We provide an overview figure for our model in the main paper. In this document, we provide a more detailed figure of our model in Figure A.2. Please see the figure and its caption for more details. We also provide a model sample in

Figure A.1. **Recovering missing information from a multi-view sequence.** Our model is flexible in that it can condition on any available information and recover the missing regions. We mark the conditioning as "Masked images", "Masked origins", and "Masked directions". The yellow regions are where information is not known. Given these conditions, we can recover all the missing information in the "Inpainted" rows. Notice that we are recovering fully unknown images, unknown poses, partial images, and we are generating two fully new images and poses at the same time (far right). The ground truth "GT" rows are provided as reference from the original capture, but the model only needs to follow the GT when the input provides the appropriate information. Our model does not require a "GT" reference and instead the model can be used to complete casually captured scenes where there is no reference, as shown in our paper. The origins and directions are all within the cube $[-1, 1]^3$ so can be visualized in RGB space.

Figure A.1 showing a combination of known images, missing images, missing poses, partial images, and where both images and poses are unknown, resulting in unconditional generations.

D.3. Raymap Prediction Discussion

We solve for camera poses with a MultiDiffusion-style approach [33, 61], where we use a smaller sequence size as input to the model and average predictions in order to predict a larger context size. We found this approach to give higher quality results for raymap recovery compared to passing in all images together in the same forward-pass. However, this finding is not true for generating image content. For the image prediction task, it's actually better to pass in all images in the same forward-pass. We suspect this finding is because relative camera pose estimation will be unaffected by cameras that do not look in the same areas, but image generation will still be influenced by all context in the scene.

D.4. Inference Speed

We provide inference speeds in Table A.1. We report this on a NVIDIA A100. Here we sample for 50 steps, but in practice one could sample for 24 steps and achieve similar quality results, while halving the inference time. Our model is on the order of seconds to generate a handful of multi-view images, which makes it useful to use with open-sourced reconstruction frameworks [49].

Table A.1. **Inference time of our model.** Measured for 50 denoising steps, including time for VAE encoding and decoding.

Resolution	4 Images	8 Images	16 Images	32 Images
256×256	9 sec	9 sec	12 sec	23 sec
512×512	12 sec	23 sec	53 sec	2 min

E. Evaluation Details

E.1. CAT3D-Sequence-Size Baseline

For this baseline, we choose 3 images from a casual capture and use them for conditioning. Then, we generate 20 rounds

of inpaints, generating 6 new views each time to reach a total of 120 new generations, the same number that we generate with our method that uses larger sequence sizes. The 6 target views are sampled at a random elevation and height on a cylinder. In practice, CAT3D [11] uses 5 new views, but we use 6 to ease comparison in our implementation.

Figure A.3. **Normal regularization.** We find that aligning depth-derived normals with 3D Gaussian Splatting normals helps improve our reconstructions.

E.2. Relative Rotation Accuracy Metric

We use LoFTR [48] (outdoor weights) and Kornia [38] to recover the relative rotation between frames sampled within 1 second of each other (in our 10-second videos). We sample 20 pairs per capture and report the average Relative Rotation Accuracy for our baselines.

E.3. Nerfbusters Dataset

We use the training splits of the Nerfbusters dataset [61]. Nerfbusters was captured with a training and evaluation video, but we only consider the training video, since it mimics the casual capture of an inexperienced photogrammetry user.

E.4. NeRFiller Dataset

We obtain the NeRFiller dataset and camera paths from the NeRFiller project [61]. We exclude the "backpack" capture for our analysis because it failed to reconstruct with Splatfacto [49, 64]. We suspect this is because the "backpack" capture is a forward-facing scene originally from SPIn-NeRF [32].

F. Normal Regularization

Our normal regularization loss $\mathcal{L}_{\text{align}} = \|\text{sg}(N_{\text{r}}) - N_{\text{d}}\|_2^2 + \|\text{sg}(N_{\text{r}}) - N_{\text{d}}\|_2^2$ aligns 3D Gaussian normals with depth-derived normals. Specifically, N_{r} are rendered normals from 3D Gaussians, oriented towards the camera, and N_{d} are normals derived from rendered depth maps by backprojecting rendered depth and computing cross products for nearby points. We found that applying this regularization at the start of training can prevent the Gaussians from moving around gracefully during the start of optimization, however, if we start the regularization after 10K steps when the initial structure settles, we can significantly improve geometry. Figure A.3 shows our results with and without normal regularization. Note that we also use a TV loss to help create smoother surfaces rather than jagged ones, but it has little effect on the geometry overall.
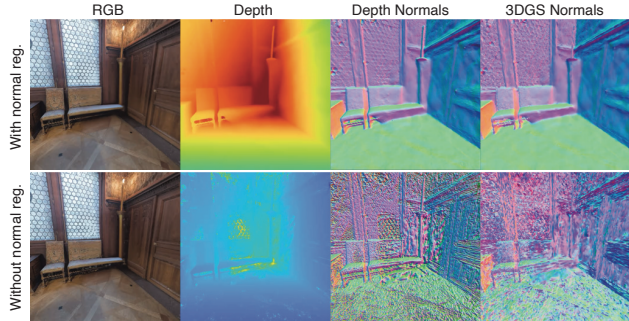
Multi-View Inputs

Images | Rays | Image masks | Ray masks
I_i | R_i | $\mathcal{M}_i^{\mathrm{I}}$ | $\mathcal{M}_i^{\mathrm{R}}$

VAE Compression and Adding Noise

Noisy image latents $\tilde{z}_{i,t}^{\mathrm{I}}$ | Noisy ray latents $\tilde{z}_{i,t}^{\mathrm{R}}$ | Known image latents $\mathcal{E}^{\mathrm{I}}(I_i \odot \mathcal{M}_i^{\mathrm{I}})$ | Known ray latents $\mathcal{E}^{\mathrm{R}}(R_i \odot \mathcal{M}_i^{\mathrm{R}})$ | Known image masks $\mathcal{D}(\mathcal{M}_i^{\mathrm{I}})$ | Known ray masks $\mathcal{D}(\mathcal{M}_i^{\mathrm{R}})$

$\tilde{z}_{1,t}^{\mathrm{I}}$ $\tilde{z}_{2,t}^{\mathrm{I}}$ ... $\tilde{z}_{N,t}^{\mathrm{I}}$

Noisy image latent, noisy ray latents, ... [conditioning]

$s_{1,t}$ $s_{2,t}$ $s_{N,t}$ ...

Patches and Embeddings

$$p' = \psi_{\mathrm{2D}}(p) + \psi_{\mathrm{Idx}}(p)$$

Transformer Block (x24)

Ada LayerNorm Continuous

Multi-Head Self-Attention

Ada LayerNorm Continuous

Multi-Head Cross-Attention

Time Conditioning

Linear | SiLU | Linear | Frequency ← Timestep

Text Conditioning

Text → Encoder

Project to correct dimensions and rearrange

Ada LayerNorm Continuous

Linear

Rearrange

Denoised Images and Rays

$\{z_1^{\mathrm{I}}, z_1^{\mathrm{R}}\}$

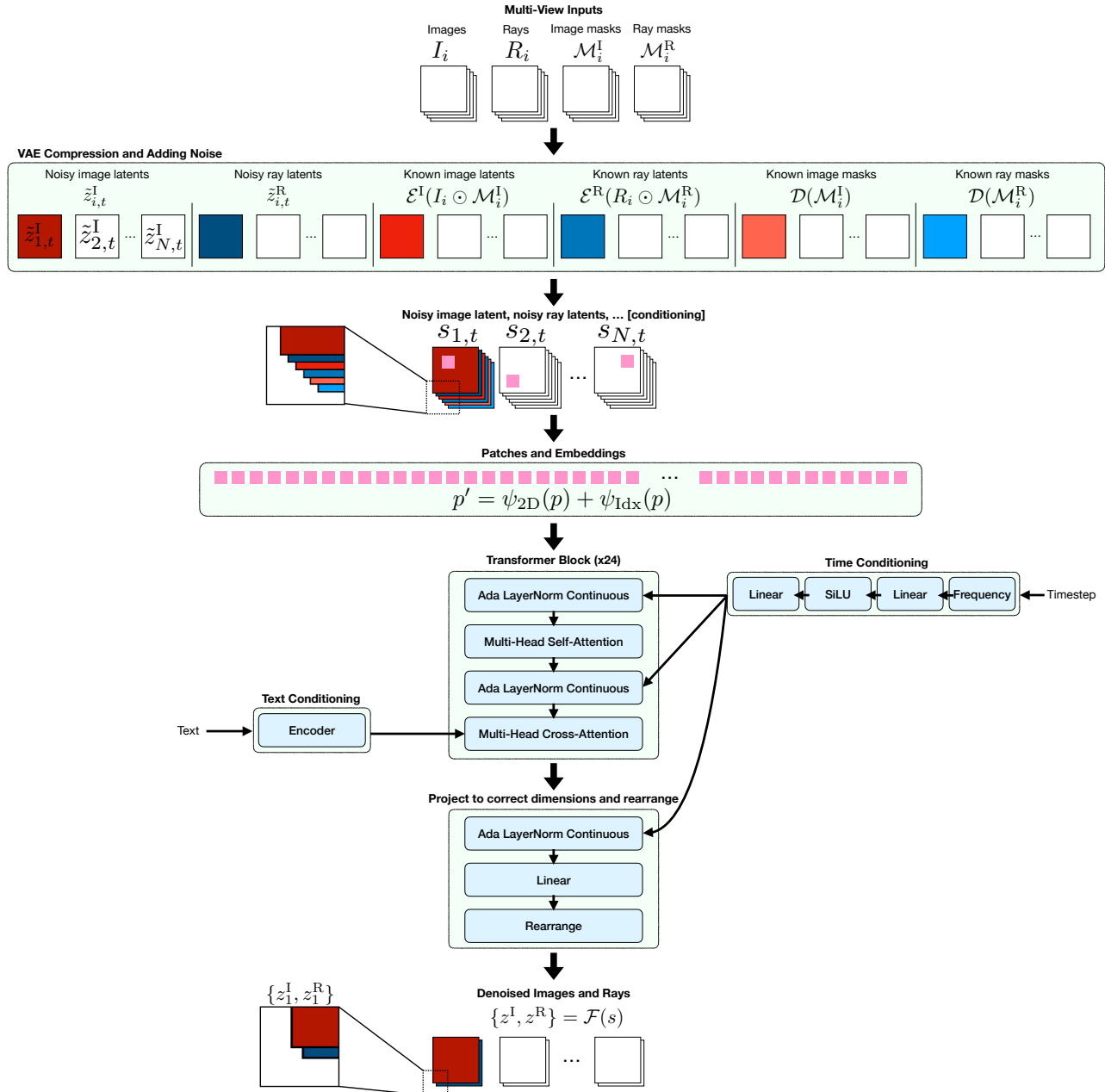$$\{z^{\mathrm{I}}, z^{\mathrm{R}}\} = \mathcal{F}(s)$$

Figure A.2. **Fillerbuster model architecture.** Here we show the full model architecture from input (top) to output (bottom). We use the same notation as the main paper, where I denotes images, R denotes raymaps, and $\mathcal{M}$ denotes masks indicating where we know information or not. We inject various conditioning time and text conditioning into the model, as shown on the sides of the transformer. Timestep conditioning is necessary because our model is a diffusion transformer. Text, however, is optional. We include it to jointly train on our single-view image collection, which has text annotations, whereas the multi-view sequences are always trained with an empty string as the text prompt. The "Ada LayerNorm Continuous" is normalization with scale and shift modulation from the continuous time conditioning.