A Limitations, Future Work and Societal Impact

Although GAUDI represents a step forward in generative models for 3D scenes, we would like to clearly discuss the limitations. One current limitation of our model is the fact that inference is not real-time. The reason for this is two fold: (i) sampling from the DDPM prior is slow even if it is amortized for the whole 3D scene. Techniques for improving inference efficiency in DDPMs have been recently proposed [63, 71, 72] and can complement GAUDI. (ii) Rendering from a radiance field is not as efficient as rendering other 3D structures like meshes. Recent work have also tackled this problem [28, 73, 48] and could be applied to our approach. In addition, many of the latest image generative models [45, 36, 56] use multiple stages of up-sampling through diffusion models to render high-res images. These up-sample stages could be directly applied to GAUDI. In addition, one could considering studying efficient encoders to replace the optimization process to find latents. While attempts have been made at using transformers [57] for short trajectories (5-10 frames) it is unclear how to scale to thousands of images per trajectory like the ones in [1]. One additional limitation is to tackle infinitely large or "boundless" indoor or outdoor scenes. The current formulation of GAUDI assumes a physical volume in space (defined by the physical size of each element in the tri-plane latent $\mathbf{W}$) on which radiance is defined. When the camera moves outside that space there's no defined radiance to render images (*e.g.* similar to the case of meshes). To tackle the case in which one wants to model a infinitely large scene one could make both the camera pose and the radiance field representation $\mathbf{W}$ be a function of time step embedding s. This will allow for the radiance field to change as the camera moves. Finally, the main limitation for a model like GAUDI to exhibit improved generation and generalization abilities is the lack of massive-scale and open-domain 3D datasets. In particular ones with other associated modalities like textual descriptions.

When considering societal impact of generative models a few aspects that need attention are the use generative models for creating disingenuous data, *e.g.* "DeepFakes" [34], training data leakage and privacy [66], and amplification of the biases present in training data [20]. One specific ethical consideration that applies to GAUDI is the impact that a model which can easily create immersive 3D scenes can have on future generations and their detachment of reality [3]. For an in-depth review of ethical considerations in generative modeling we refer the reader to [55].

B Experimental Settings and Details

In this section we describe details about data and model hyper-parameters. For all experiments our latents $\mathbf{z}_{\text{scene}}$ and $\mathbf{z}_{\text{pose}}$ have 2048 dimensions. In the first stage, when latents are optimized via Eq. 2, $\mathbf{z}_{\text{scene}}$ gets reshaped to a $8 \times 8 \times 32$ feature map before feeding it to the scene decoder network. In the second stage, when training the DDPM prior we reshape $\mathbf{z}_{\text{scene}}$ and $\mathbf{z}_{\text{pose}}$ to $8 \times 8 \times 64$ latent and leverage the power of a UNet [54] denoising architecture.

For each dataset, trajectories have different length, physical scale, as well as near and far planes for rendering, which we adjust accordingly in our model.

Vizdoom [23]: In Vizdoom, trajectories contains 600 steps on average. In each step the camera is allowed to move forward 0.5 game units or rotate left or right by 30 degrees. We set the unit length of an element in the tri-plane representation as 0.05 game units (meaning each latent code $\mathbf{w}_{xyz}$ represents a volume of space of 0.05 cubic game units). The near plane is at 0.0 game units and the far plane at 800 game units. We use the data and splits provided by [8].

Replica [64]: In Replica, all trajectories contain 100 steps. In each step, the camera can either rotate left or right by 25 degrees or move forward 15 centimeters. We set the unit length of an element in the tri-plane representation as 25 centimeters (meaning each latent code $\mathbf{w}_{xyz}$ represents a volume of space of 0.25 cubic centimeters). The near plane is at 0.0 meters and the far plane at 6 meters. We use the data and splits provided by [8].

VLN-CE [26]: in VLN-CE trajectories contain a variable number of steps between 30 and 150, approximately. In each step, the camera can either rotate left or right by 25 degrees or move forward 15 centimeters. We set the unit length of an element in the tri-plane representation as 50 centimeters.

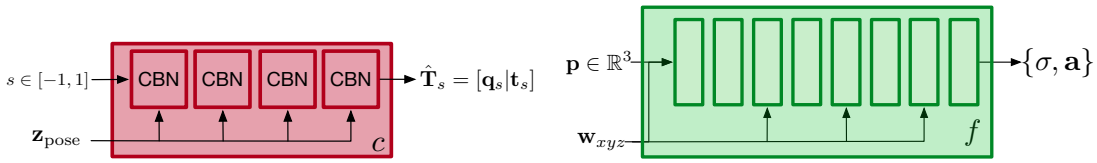

Figure 10: (a) Architecture of the camera pose decoder network. (b) Architecture of the radiance field network.

The near plane is at 0.0 meters and the far plane at 12 meters. We use the data and training splits provided by [26].

ARKitScenes [1]: in ARKitScenes trajectories contain a number of steps around 1000 on average. In these trajectories the camera is able to move continuously in any direction and orientation. We set the unit length of an element in the tri-plane representation as 20 centimeters. The near plane is at 0.0 meters and the far plane at 8 meters. We use the 3DOD split of data provided by [1]

C Decoder Architecture Design and Details

In this section we describe the decoder model in Fig. 2 in the main paper. The decoder network is composed of 3 modules: **scene decoder**, **camera pose decoder** and **radiance field decoder**.

- The **scene decoder** network follows the architecture of the VQGAN decoder [11], parameterized with convolutional architecture that contains a self-attention layers at the end of each block. The output of the scene decoder is a feature map of shape $64 \times 64 \times 768$. To obtain the tri-plane representation $\mathbf{W} = [\mathbf{W}_{xy}, \mathbf{W}_{xz}, \mathbf{W}_{yz}]$ we split the channel dimension of the output feature map in 3 chunks of equal size $64 \times 64 \times 256$.
- The **camera pose decoder** is implemented as an MLP with 4 conditional batch normalization (CBN) blocks with residual connections and hidden size of 256, as in [31]. The conditional batch normalization parameters are predicted from $\mathbf{z}_{\text{pose}}$. We apply positional encoding to the inputs the camera pose encoder ($s \in [-1, 1]$). Fig. 10(a) shows the architecture of the camera pose decoder module.
- The **radiance field decoder** is implemented as an MLP with 8 linear layers with hidden dimension of 512 and LeakyReLU activations. We apply positional encoding to the inputs the radiance field decoder ($\mathbf{p} \in \mathbb{R}^3$) and concatenate the conditioning variable $\mathbf{w}_{xyz}$ to the output of every other layer in the MLP starting from the input layer (*e.g.* layers 0, 2, 4, and 6). To improve efficiency, we render a small resolution feature map of 512 channels (two times smaller than the output resolution) instead of an RGB image and use a UNet [54] with additional deconvolution layers to predict the final image [8, 38]. Fig. 10(b) shows the architecture of the radiance field decoder module.

For training we initialize all latents $\mathbf{z} = 0$ and train them jointly with the parameters of the 3 modules. We use the Adam optimizer and a learning rate of 0.001 for latents and 0.0001 for model parameters. We train our model on 8 A100 NVIDIA GPUs for 2-7 days (depending on dataset size), with a batch size of 16 trajectories where we randomly sample 2 images per trajectory.

D Prior Architecture Design and Details

We employ a Denoising Diffusion Probabilistic Model (DDPM) [17] to learn the distribution $p(Z)$. Specifically, we adopt the UNet architecture from [37] to denoise the latent at each timestep. During training, we sample $t \in \{1, ..., T\}$ uniformly and take the gradient descent step on θ_p from Eq. 3. Different from [37], we keep the original DDPM training scheme with fixed time-dependent covariance matrix and linear noise schedule. During inference, we start from sampling latent from zero-mean unit-variance Gaussian distribution and perform the denoising step iteratively. To accelerate the sampling efficiency, we leverage DDIM [63] to denoise only 50 steps by modeling the deterministic non-Markovian diffusion processes.

For conditional generative modelling tasks, the conditioning mechanism should be general to support conditioning inputs from diverse modalities (i.e. text, image, categorical class, etc.). To fulfill this requirement, we first project the conditional inputs into an embedding representations $\mathbf{c}$ via a modality-specific encoder. For text conditioning, we employ a pre-trained RoBERTa-base [29]. For image conditioning, we employ a ResNet-18 [15] pre-trained on ImageNet. For categorical conditioning, we employ a trainable per-environment embedding layer. We freeze the encoders for text and image inputs to avoid over-fitting issues. We borrow the cross attention module from LDM [52] to fuse the conditioning representation $\mathbf{c}$ with the intermediate activations at multiple levels in the UNet [54]. The cross-attention module implements an attention mechanism with key and value generated from $\mathbf{c}$ while the query generated from the intermediate activations in the UNet architecture (we refer readers to [52] for more details).

For training the DDPM prior, we use the Adam optimizer and learning rate of $4.0e^{-06}$. We train our model on 1 A100 NVIDIA GPU for 1-3 days for unconditional prior learning and 3-5 days for conditional prior learning experiments (depending on dataset size), with a batch size of 256 and 32 respectively. For the hyper-parameters of the DDPM model, we set the number diffusion steps to 1000, noise schedule as linearly decreasing from 0.0195 to 0.0015, base channel size to 224, attention resolutions at [8, 4, 2, 1], and number of attention heads to 8.

Inference of the radiance field is amortized across the whole scene. This means that we only need to sample from the prior once and then we can render as many views of the scene as needed. Sampling from the prior takes 1.52s. Once we obtain the scene embedding, per frame rendering takes $1.6s$ at 128x128 resolution. These results are obtained on a single A100 NVIDIA GPU.

E Ablation Study

We now provide additional ablations studies for the critical components in GAUDI. First, we analyze how the dimensionality of the latent code z_d and the magnitude of β affect the optimization problem defined in Eq. 2. Tab. 4 shows reconstruction metrics for both RGB images and camera poses for a subset of 100 trajectories in the VLN-CE dataset [26]. We observe a clear trend where increasing the magnitude of β makes it harder to find latent codes with high reconstruction accuracy. This drop in accuracy is expected since β controls the amount of noise in latent codes during training. Finally, we observe that reconstruction performance starts to degrade when the latent code dimensionality grows past 2048.

		$l_1 \downarrow$	PSNR ↑	SSIM ↑	Rot Err. ↓	Trans. Err ↓
$\beta = 0.1$	$z_d = 2048$	7.63e-3	39.12	0.984	4.61e-3	2.90e-3
$\beta = 0.1$	$z_d = 4096$	7.89e-3	38.55	0.982	4.91e-3	2.76e-3
$\beta = 0.1$	$z_d = 8192$	9.02e-3	36.33	0.978	5.62e-3	3.36e-3
$\beta = 1.0$	$z_d = 2048$	1.00e-2	34.82	0.972	6.32e-3	3.77e-3
$\beta = 1.0$	$z_d = 4096$	1.11e-2	34.46	0.965	7.27e-3	5.69e-3
$\beta = 1.0$	$z_d = 8192$	1.54e-2	32.28	0.916	1.11e-2	7.13e-3
$\beta = 10.0$	$z_d = 8192$	3.89e-2	24.89	0.799	7.59e-2	3.61e-2
$\beta = 10.0$	$z_d = 4096$	9.25e-2	17.52	0.499	1.56e-1	6.30e-2
$\beta = 10.0$	$z_d = 2048$	1.35e-1	12.74	0.275	5.25e-1	1.29e-1

Table 4: Ablation experiment for the critical parameters of the optimization process described in Eq. 2

In addition, we also provide ablation experiments for the second stage of our model where we learn the prior $p(Z)$. In particular, we ablate critical factors of our model: the importance of learning corresponding scene and pose latents, the width of the denoising network in the DDPM prior, and the noise scale parameter β. In Tab. 5 we show results for each factor. In particular, in the first two rows of Tab. 5 we show the result of training the prior while breaking the correspondence of $\mathbf{z} = [\mathbf{z}_{\text{pose}}, \mathbf{z}_{\text{scene}}]$. We break this correspondence by forming random pairs of $\mathbf{z} = [\mathbf{z}_{\text{pose}}, \mathbf{z}_{\text{scene}}]$ after optimizing the latent representations, and then training the prior on these random pairs. We observe that training the prior to render scenes from a random pose latent impacts both the FID and SwAV-FID scores substantially, which provides support for our claim that the distribution of valid camera poses depends on the scene. In addition, we can see how the width of the denoising model affects performance. By increasing the number of channels, the DDPM prior is able to better capture

the distribution of latents. Finally, we also show how different noise scales β impact the capacity of the generative model to capture the distribution of scenes. All results Tab 5 are performed on the full VLN-CE dataset [26].

	VLN-CE [26]	
	FID ↓	SwAV-FID ↓
GAUDI	18.52	3.63
GAUDI w. Random Pose	83.66	10.73
Base Channel Size = 64	104.27	13.21
Base Channel Size = 128	22.04	4.35
Base Channel Size = 192	18.61	3.79
Base Channel Size = 224	18.52	3.63
Noise Scale β = 0.0	18.48	3.68
Noise Scale β = 0.1 (same as 1st stage)	18.52	3.63
Noise Scale β = 0.2	18.48	3.67
Noise Scale β = 0.5	20.20	4.11

Table 5: Ablation study for different design choices of GAUDI.

In Tab. 6 we report ablation results for modulation on the denoising architecture in the DDPM prior. We compare cross-attention style conditioning as in LDM [52] with FiLM style conditioning [42]. For FiLM style conditioning, we take the mean of the conditioning representation $\mathbf{c}$ across spatial dimension and project it into the same space as denoising timestep embedding. After that, we take the sum of the conditioning and timestep embedding and predict the scaling and shift factors of the affine transformation applied to the UNet intermediate activations. We compare the performance of the two conditioning mechanisms in Tab. 6. We observe that the cross-attention style conditioning performs better than the FiLM style across all our conditional generative modeling experiments.

	Text Conditioning		Image Conditioning		Categorical Conditioning	
	FID ↓	SwAV-FID ↓	FID ↓	SwAV-FID ↓	FID ↓	SwAV-FID ↓
FiLM Module [42]	20.99	4.11	21.01	4.21	18.75	3.63
Cross Attention [52]	18.50	3.75	19.51	3.93	18.74	3.61

Table 6: Ablation study for conditioning mechanism of GAUDI.

F Arbitrary viewpoint synthesis

In this section, we provide empirical evidence to support the claim that GAUDI allows for arbitrary viewpoint synthesis and does not overfit to the camera poses seen during training. In order to do this, we take a model trained on VLN-CE [26] and modify the inference process.

Specifically, during inference, we sample latents $\mathbf{z}_{\text{pose}}$ and $\mathbf{z}_{\text{scene}}$ and decode camera poses. Before rendering images encoded in the radiance field in $\mathbf{z}_{\text{scene}}$ we perturb the decoded camera poses by adding uniform translation noise in the horizontal plane (up to $50\,\text{cm}$) and uniform rotation noise in the yaw axis (up to $20°$). We expect that as camera poses are perturbed with increasing noise, the FID score will also increase, this is because camera poses might move to non-valid navigable areas of the scene (*e.g.* . outside of a hallway). In Tab. 7 we report the FID scores of GAUDI with perturbed camera poses. Our results show that GAUDI is reasonably robust to noise in the camera poses and while FID does indeed increase as the cameras are perturbed, GAUDI still generates realistic images, outperforming baselines by a wide margin.

G Trajectory level metrics

In order to provide a comprehensive quantitative analysis of the samples generated by GAUDI we perform an empirical analysis at the trajectory level. This analysis is meant to shed light on the temporal and multi-view consistency of the trajectories generated by GAUDI. In order to do this, we compute FVD scores [67] on a set of approximately 600 trajectories in VLN-CE obtaining a score of 143.53. To put this number into context, DriveGAN [24] a recent model for video prediction obtains

	VLN-CE [26]	
	FID ↓	SwAV-FID ↓
π-GAN[6]	90.43	8.65
GRAF [60]	151.26	14.07
GSN [8]	43.32	6.19
GAUDI	**18.52**	**3.63**
trans noise $U(0, 25\text{ cm})$, rot noise $U(0, 10^\circ)$	20.38	4.01
trans noise $U(0, 50\text{ cm})$, rot noise $U(0, 20^\circ)$	25.90	4.68

Table 7: Arbitrary view synthesis results on VLN-CE.

an FVD score of 360.00 on Gibson (a dataset of indoor scenes that is very similar to VLN-CE). Computing the FVD score from the ground truth training trajectories to themselves results in a score of 43.09, this number serves as the lower bound in terms of FVD score that a model could obtain.

In order to calculate the FVD score, we obtain clips of 20 equidistant frames from all trajectories sampled from our model as well as randomly sampled ground truth trajectories. We then use the open implementation of FVD `https://github.com/google-research/google-research/tree/master/frechet_video_distance` to calculate the FVD score. Based on recent approaches for video generative modeling [67, 4, 30, 24], the FVD score of 143.53 indicates that GAUDI is able to capture multi-view/temporal consistency as shown in the qualitative video samples provided in the appendix.

H Additional Visualizations

In this section we provide additional visualizations for both figures in this appendix and videos that can be found attached in the supplementary material. In Fig. 11 we provide additional interpolations between random pairs of latents obtained for VLN-CE dataset [26], where each row represents a interpolation path between a random pair of latents (*i.e.* rightmost and leftmost columns). We can see how the model tends to produce smoothly changing interpolation paths which align similar scene content. In addition we refer readers to the folder `./interpolations` in which videos of interpolations can be found where for each interpolated scene we immersively navigate it by moving the camera forwards and rotating left and right.

In addition, we provide more visualization of samples from the unconditional GAUDI model in Fig. 12 for VLN-CE [26], Fig. 13 for ARKitScenes [1] and Fig. 14 for Replica [64]. In all these figures, each row represents a sample from the prior that is rendered from its corresponding sampled camera path. We note how these qualitative results reinforce the fidelity and variability of the distribution captured by GAUDI, which is also reflected in the quantitative results in Tab. 2 of the main paper. In addition, the folder `./uncond_samples` contains videos of more samples from the unconditional GAUDI model for all datasets.

Finally, the folder `./cond_samples` contains a video showing samples from GAUDI conditioned on different modalities like text, images or categorical variables. These visualizations corresponds to the results in Sect. 4.5 of the main paper.

I License

Due to licensing issues we cannot release the VLN-CE [26] raw trajectory data and we refer the reader to `https://github.com/jacobkrantz/VLN-CE` and to the license of the Matterport3D data `http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf`.

Figure 11: Additional interpolation of 3D scenes in latent space for the VLN-CE dataset [26]. Each row corresponds to a different interpolation path between random pairs of latent representations $(\mathbf{z}_i, \mathbf{z}_j)$.

Figure 12: Additional visualizations of scenes sampled from unconditional GAUDI for VLN-CE dataset [26]. Each row to a scene rendered from camera poses sampled from the prior.

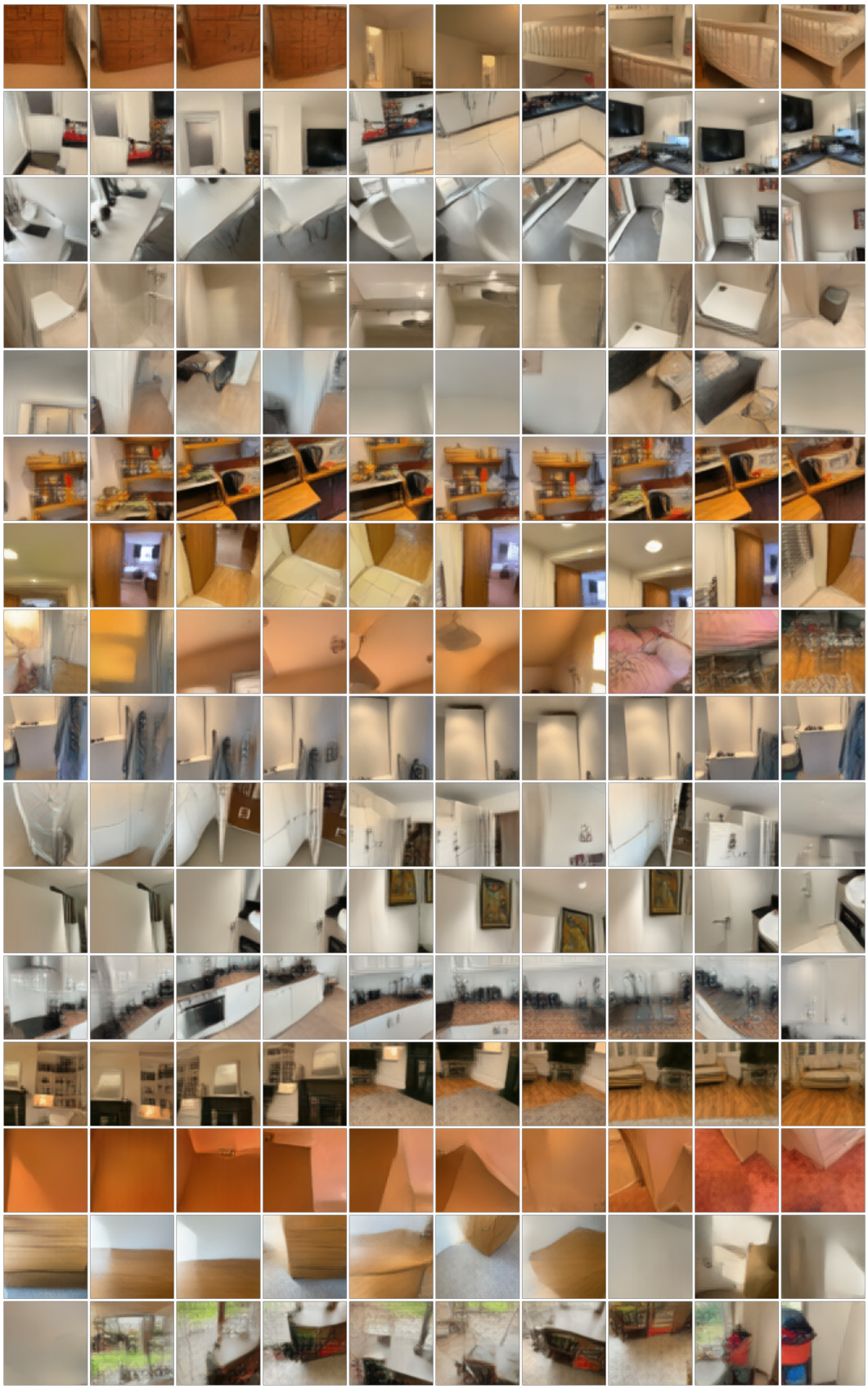

Figure 13: Additional visualizations of scenes sampled from unconditional GAUDI for ARKitScenes dataset [1]. Each row corresponds to a scene rendered from camera poses sampled from the prior.

Figure 14: Additional visualizations of scenes sampled from unconditional GAUDI for Replica dataset [64]. Each row corresponds to a scene rendered from camera poses sampled from the prior.