

Depth Scaling for Diffusion Edits

Muhaimen Shamsi, and Jadyn Wall

Abstract—We address perspective mismatch when translating objects within a single image using diffusion-based editing. Existing training-free methods move objects in the image plane but leave their apparent scale unchanged, producing unrealistic composites. We propose a depth-aware pipeline that rescales the source object before reinsertion. A monocular depth estimator provides absolute depths at the source and target locations, and a scale factor is derived from their ratio. The object is removed with inpainting, resized, and reinserted with depth-conditioned diffusion guidance. Compared to a disparity-only baseline and a no-scaling variant, our approach increases Mask IoU (0.649 vs. 0.601 and 0.629) while maintaining competitive SSIM (0.842) and FID (44.5). Qualitatively, absolute-depth yields the most plausible placement and identity preservation, whereas disparity and no-scaling suffer from over/under-scaling and color shifts. Our results indicate that depth scaling materially improves object identity and perspective during relocation without retraining diffusion backbones.

Index Terms—Depth-aware image editing, Diffusion models, Object-level manipulation, Perspective-consistent scaling

1 INTRODUCTION

EDITING objects in images without relying on 3D priors has gained significant research attention with the advent of diffusion models. These models enable fine-grained control over visual content, allowing users to modify existing images while preserving overall scene coherence. Applications span data augmentation for robotic simulations, and virtual product previews. Recent work demonstrates increasing control over structural edits such as rotation, scaling, and spatial relocation, as well as stylistic edits including relighting and texture adjustment. Building on this progress, our project focuses on improving geometric consistency during structural edits. Specifically, we aim to use depth priors to preserve an object’s apparent scale as it is repositioned in a scene, enabling perspective-correct manipulation in single-image editing tasks.

2 RELATED WORK

Image Editing with Diffusion Models. Many recent approaches enable image editing using training-free diffusion models. This proposal focuses on the most relevant and recent methods for object translation and scaling with depth awareness.

DiffUHaul [1], FreeFine [2], and SceneDiffusion [3] each demonstrate the ability to translate objects within a scene using pretrained diffusion models, though their methods differ in implementation. The limitation shared by these approaches is that object movement occurs purely within the 2D image plane, meaning relocated objects are not rescaled according to their new depth positions, leading to inconsistencies in perspective and spatial realism.

Diffusion Handles [4] introduces a framework for 3D-aware object editing by lifting diffusion model activations into a three-dimensional latent space, achieving geometric consistency without explicitly solving the inverse graphics problem.

Object Insertion. Zero-Shot Depth-Aware Image Editing with Diffusion Models (ZSDAD) [5] proposes two methods, DeGLaD and FeatGLaC, for depth-conditioned object insertion, allowing new objects to be placed at specific scene depths.

DeGLaD - Depth Guided Layer Decomposition - separates an image into a foreground and background layer based on an input depth value.

FeatGLaC - Feature Guided Layer Compositing - composites image layers with a pretrained diffusion model to achieve realistic lighting and colour consistency. ZSDAD [5] does not support intra-image object transformations, such as moving an existing object to a new position while maintaining scale and occlusion consistency. However, it provides a foundation for our objective.

The proposed project builds upon this line of work by integrating depth-conditioned diffusion editing with training-free geometric transformations, thereby adding the capability for realistic, depth-aware object repositioning within a single image.

3 METHOD

Our goal is to reposition an object within a single RGB image while preserving its apparent scale according to scene depth. Given an input image x , a source pixel location, and a target pixel location, our pipeline extracts the object at the source, estimates the depth at both locations, predicts the appropriate scale factor, removes the object from the background, and reinserts a scaled version of the object at the target location using depth-conditioned diffusion editing. Figure 1 provides an overview.

3.1 Preliminaries

Latent Diffusion Models. Diffusion models [6, 7] can generate an image x_0 by starting from pure Gaussian noise x_T and iteratively denoising with a U-Net ϵ_θ . To reduce computational cost, latent diffusion models [8] perform denoising in lower dimensional latent space and use a variational autoencoder to transition between the pixel and latent space.

• M. Shamsi and J. Wall are with the Department of Computer Science, University of Toronto.

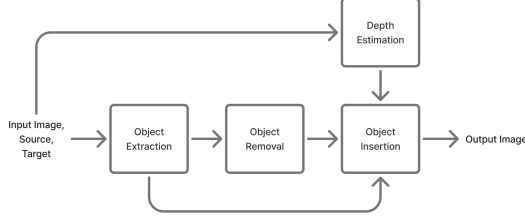


Fig. 1: Proposed depth aware editing pipeline.

The latent representation of an image at timestep t is z_t and each U-Net feature in layer i is represented as $\Psi_{i,t}$. This notation is important to understand Object Insertion.

Guidance. In diffusion models, guidance is used to lead the denoising network towards a desired outcome with a guidance function $\mathcal{G}$:

$$\tilde{\epsilon}_\theta(z_t, t) = \epsilon_\theta(z_t, t) + \lambda \nabla_{z_t} \mathcal{G}(z_t, t) \quad (1)$$

where λ is the guidance scale. This adjusts the denoiser's prediction to steer the image generation.

Null-Text Inversion. Editing real images requires an inversion procedure that reconstructs the original image from a noisy latent. Null-text inversion [9] is the method used in FeatGLaC [5], which recursively optimizes the unconditional embedding $\varnothing_t$ so that the DDIM [7] denoising trajectory reproduces the input image. At each timestep:

$$\tilde{\epsilon}_\theta(z_t, t, c, \varnothing_t) = w \cdot \epsilon_\theta(z_t, t, c) + (1 - w) \cdot \epsilon_\theta(z_t, t, \varnothing_t) \quad (2)$$

$$\varnothing_t \leftarrow \varnothing_t - \eta \nabla_{\varnothing_t} \|z_{t-1}^* - z_{t-1}(\tilde{z}_t, \varnothing_t, c)\|_2^2 \quad (3)$$

$$\tilde{z}_{t-1} \leftarrow z_{t-1}(\tilde{z}_{t-1}, \varnothing_t, c), \varnothing_{t-1} \leftarrow \varnothing_t \quad (4)$$

where z_t^* is the latent obtained by deterministically noising the input image. The optimized $\{\varnothing_t\}$ and terminal latent $\tilde{z}_T$ are output from the null-text inversion to be used when editing the image in latent space.

3.2 Depth Estimation

We use the pretrained monocular depth estimator Depth Anything [10] to get the depth/disparity map of the input image. Given the source and target pixel locations, we extract depths d and d' , respectively to calculate the scale factor s applied to the repositioned object:

$$s = \frac{d}{d'} \quad (5)$$

which ensures that the object's apparent size matches its new depth.

3.3 Object Extraction

We use the segmentation model Segment Anything Model [11] to get a mask of the object at the source location. The mask is then applied to the input image to obtain an RGBA cutout of the object.

3.4 Object Removal

We remove the object by inpainting the source region using Stable Diffusion 2.1 [8]. To mitigate residual artifacts, we dilate the mask and blur its edges prior to inpainting. We guide the inpainting process with the prompt:

high quality, high resolution, indoor scene

and use the negative prompt:

object

This step helps in producing an artifact-free image suitable for object reinsertion.

3.5 Object Insertion

To insert the object at the target location while accounting for the change in depth, we use the Zero-Shot Depth-Aware Editing (ZSDAD) framework [5], which introduces two complementary methods: DeGLaD for depth-conditioned layer decomposition and FeatGLaC for feature-space compositing. An overview is provided in Figure 2.

Layer Decomposition (DeGLaD). Given the target depth d' , DeGLaD decomposes the image into foreground and background layers at that depth with masks M_{fg} and M_{bg} , respectively.

Feature-Guided Layer Compositing (FeatGLaC). Direct pixel-space compositing suffers from poor realism due to changes in lighting conditions at the object's new location, while early latent-space compositing when using a diffusion model for inpainting can lead to loss of object identity. Figure 3 shows the difference in outcomes between these compositing methods. FeatGLaC addresses both issues by progressively blending the foreground and background at multiple denoising steps.

The pretrained inpainting model AnyDoor [12] is the diffusion model used in FeatGLaC to create the final output image.

At timestep t , with U-Net features $\Psi_{i,t}$ and edited image features $\Psi_{i,t}^e$, FeatGLaC forces the foreground of the composited image to be similar to the foreground layer obtained from DeGLaD:

$$\mathcal{G} = \sum_i \|M_{fg} * (\Psi_{i,t} - \Psi_{i,t}^e)\|^2 \quad (6)$$

and updates latent z_t^e via:

$$\tilde{z}_t^e = z_t^e - \nabla_{z_t^e} \mathcal{G} \quad (7)$$

We apply this energy for denoising steps $t = 30 - 50$, repeating the update three times per step to maintain object identity, geometry, and shading consistency.

3.6 Implementation Details

Inputs consist of an RGB image, a source pixel, and a target pixel; the output is an RGB image with the object repositioned from the source pixel to the target pixel and scaled to reflect its new depth. We use pretrained models Depth Anything [10], Segment Anything Model [11], Stable Diffusion 2.1 [8], and AnyDoor [12]. To handle arbitrary image sizes, we extract a 512×512 crop around both the source and target regions for removal and insertion,

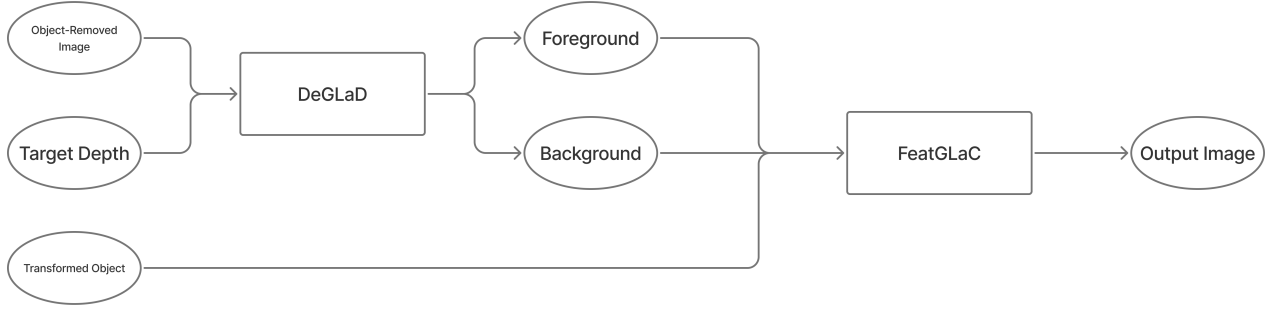


Fig. 2: Object insertion overview.

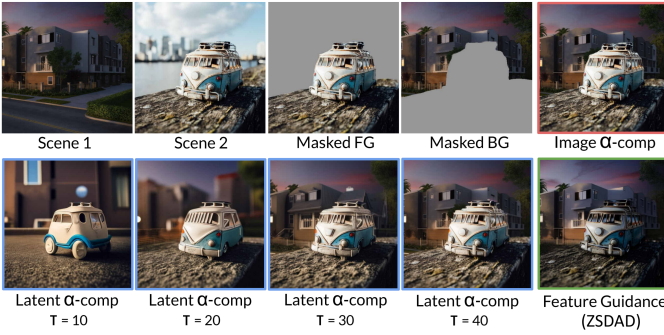


Fig. 3: **Comparing layer compositing methods:** The top right image shows pixel space compositing. The bottom row, with the exception of the first image to the right, shows latent space compositing at different denoising steps, T . The bottom right image shows layer compositing using the FeatGLaC method proposed by ZSDAD [5].

respectively. Null-text inversion [9] is performed once per image and cached for reuse, as the background crop and prompts remain fixed.

4 RESULTS

To quantify the effect of depth-aware scaling we evaluate four criteria: 1) whether the edited object lands at the intended location, 2) whether the scale matches the target depth, 3) whether fine object detail is preserved, and 4) whether the surrounding scene remains visually consistent.

4.1 Dataset

The authors of ZSDAD [5] curated a benchmark from web images to introduce depth-aware editing tasks. Their object insertion dataset consists of: 1) an RGB indoor scene image, 2) a target mask that indicates the location and size of the inserted object, 3) a scalar depth value specifying the depth of the inserted object, and 4) an RGBA image of the object to insert with a tight alpha mask.

Because this dataset is not publicly available, we curate a 35-image substitute from the MIT Indoor Scenes dataset [13]. We retain images larger than 512×512 to avoid additional resizing in our Stable Diffusion pipeline, discard frames without movable objects or with missing ground planes, and sample up to two images per category across the

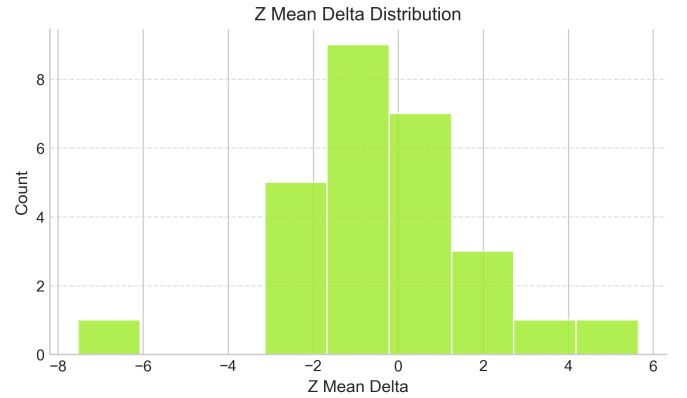


Fig. 4: Mean distance moved during object repositioning. Positive values mean the placed object appears farther away than it was in the source; negative values mean it moved closer along the camera’s z-axis.

67 indoor classes. The resulting set spans diverse lighting, composition, and clutter.

For each image we manually annotate a source and target location (pixel coordinates) and store the pairs in a single JSON file. The annotations cover a range of object scales and displacements, including objects on tables, floors, and partially occluded surfaces. Our pipeline is unable to aggregate the target object for 8 of the 35 samples, which we excluded from our results. Example failure cases are shown in Figure 5.

4.2 Repositioning

We assess target alignment by applying SAM to the edited image and querying the mask centered at the target pixel. SAM fails to return an object at the specified location for 8 of the 35 samples, indicating missed aggregation; the remaining 27 samples yield detections at the target, consistent with successful relocation under depth-aware scaling.

To quantify displacement, we compute depth maps for both the original and final images, take the mean depth inside the original object mask and the SAM-detected target mask, and subtract them to obtain the mean depth-delta. Figure 4 shows the depth-delta distribution, confirming diverse motions in the dataset.



Fig. 5: Representative samples where SAM fails to detect an object at the target point, indicating no target aggregation. Left shows the original image, right shows the pipeline output.

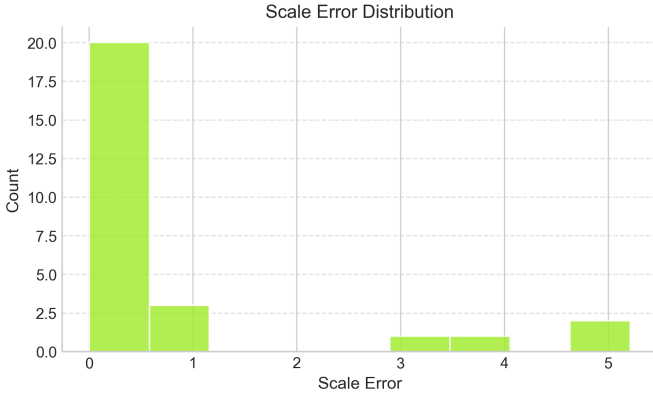


Fig. 6: Scale error, excluding failed aggregations. Values are averaged over three runs per sample.

4.3 Depth Scaling

To verify that the object has the correct scale, we use SAM to calculate the repositioned object’s mask. The target mask bounding box guiding the diffusion process estimates the expected center of the repositioned object. This center pixel is fed to SAM to extract a mask of the object’s contour. If SAM returns multiple objects, we mirror the selection algorithm used during original object extraction and choose the mask with the largest area. We then compute the ratio of the SAM masks of the original and repositioned objects to estimate the observed scaling.

$$s_{\text{obs}} = \frac{\sum M_{\text{orig}}}{\sum M_{\text{repo}}} \quad (8)$$

Figure 6 shows a histogram of scale error, which is the difference between applied and observed scale. Samples with scale error greater than 1.0 are cases with incomplete object aggregation. An example result with high scale error is shown in Figure 7.



Fig. 7: The diffusion process fails to aggregate the object at the target location, resulting in high scale error = 4.63.

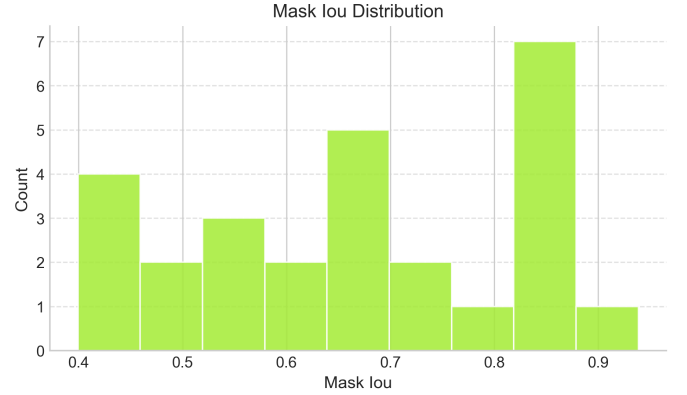


Fig. 8: Distribution of mask IoU between the SAM-extracted repositioned mask and the target mask, averaged over three runs per sample.

4.4 Object Shape

We quantify geometric fidelity by comparing the SAM mask of the repositioned object to the target mask used during conditioning. For IoU calculation, we crop the source and result images to the masked regions and resize the smaller mask to match the larger one.

(Table 1) shows that depth scaling yields the highest mean Mask IoU of 0.649, with lower values driven by samples where the repositioned object is partially or entirely missing. Figure 8 shows a histogram of Mask IoUs with a spike for samples with Mask IoU above 0.8, indicating similar object shape. Figure 9 shows the highest and lowest Mask IoUs from our experiments.

4.5 Scene Realism

Scene realism is assessed by computing SSIM and FID over the entire result image. SSIM (Structural Similarity Index Measure) measures perceived quality via luminance, contrast, and structure; its mean remains close to the baseline for both depth-scaled and unscaled repositioning (Table 1).

FID (Frechet Inception Distance) estimates the distance between distributions of Inception-v3 features for original and result images, increasing when generated images are blurry, repetitive, or unrealistic. Depth-based repositioning raises FID relative to the disparity baseline, while depth scaling produces a slightly lower (better) FID than the no-scaling configuration. An example of a high FID case is shown in Figure 10.



Fig. 9: (Top) Highest Mask IoU = 0.941, (Bottom) Lowest Mask IoU = 0.381



Fig. 10: High FID=144.70 due to major artifacts on the floor near the target.

4.6 Baseline

To compare depth scaling with the unmodified pipeline published in the ZSDAD paper [5], we run the object repositioning pipeline using disparity instead of depth maps. The original implementation uses DepthAnythingV2 but does not exploit absolute depth outputs, opting for disparity because it is faster while still enabling depth-aware object placement.

Additionally, to isolate the effects of depth scaling, we ablate it by fixing *scale factor*=1 across all conditioning streams. This baseline measures the downstream effect of scaling on FID, SSIM, and Mask IoU.

Baseline results are shown in (Table 1); the first row reports metrics using disparity maps, and the second row

metric	FID		SSIM		Mask IoU	
	mean	std	mean	std	mean	std
no scaling	50.523	41.953	0.827	0.106	0.629	0.200
disparity	31.749	31.200	0.870	0.094	0.601	0.191
abs depth	44.512	40.910	0.842	0.091	0.649	0.190

Table 1: Aggregate metrics across disparity, no scaling, and absolute-depth configurations.

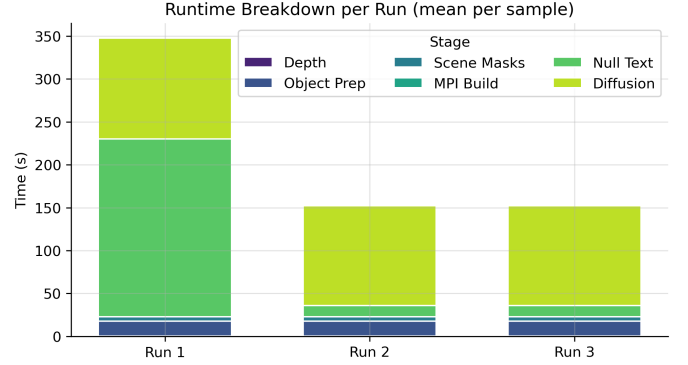


Fig. 11: High FID=144.70 due to major artifacts on the floor near the target.

reports results with depth maps but no depth scaling. Compared to using disparity maps with no scaling, our approach produces higher Mask IoU but suffers from a higher FID. There are no significant changes in SSIM between the different configurations.

4.7 Qualitative Results

Results for the no-scaling, disparity, and absolute-depth variants are shown in Appendix A. Following qualitative criteria used in diffusion-based editing work [4, 1, 3, 2, 5], we assess (1) spatial alignment and scale relative to the target layout, (2) identity and surface detail preservation, and (3) whether edits respect depth cues such as occlusions, shadows, and background integration.

Absolute-depth yields the most plausible scale and placement; no-scaling produces severely oversized objects, and disparity exhibits over- or under-scaling from poor scale estimation. Identity and fine structure are retained best with absolute depth. No-scaling and disparity lose shape fidelity on large displacements, often introducing color shifts or oversized/eroded features.

All three variants introduce minor color drift and blur; high-frequency details (e.g., shopping-cart wireframes) keep their layout but show unrealistic reflections. None of the methods handles scene physics: backgrounds are pasted through holes, shadows and reflections are missing, and lighting mismatches make objects appear composited. Cropped inpainting followed by uncropping also leaves faint box artifacts and color shifts in textured regions and faces.

4.8 Run Time

To gauge deployment viability (e.g., interior design or simulation data generation), we measure per-stage latency and report the breakdown in Figure 11. Stage costs remain stable across runs except for null-text inversion, whose latents can

be cached because they depend only on the fixed background crop, depth map, and prompt. Using fixed seeds yields deterministic inversion, enabling reuse. Caching reduces mean end-to-end time by $2.23\times$ and median time by $2.33\times$ relative to the uncached baseline, with per-sample null-text cost dropping from ~ 208 s to ~ 5 – 13 s.

5 DISCUSSION

5.1 Limitations

Our dependence on separate background and foreground masks to resolve occlusions can remove parts of large objects that span both regions, especially when the target point differs greatly in elevation from the source. Cropping for inpainting and uncropping afterwards occasionally leaves faint “ghost-box” artifacts around the edit. The pipeline script accepts only a single source/target point, so users cannot explore the full placement space. The inpainting stage to remove the original object is also prompt-sensitive and can insert unintended objects unless tuned per case. Finally, the current system is memory-intensive because it stitches together multiple third-party models, requiring at least 32GB of RAM and 24GB of VRAM.

5.2 Future Work

We plan to swap in improved predictors as they become available (e.g., newer depth, segmentation, and diffusion backbones) while keeping the pipeline unchanged. Incorporating recent work to address the brittle text prompts for inpainting [14] could make the system more robust. A more expressive interface with interactive segmentation and richer placement controls would broaden use cases. Curating a larger dataset—e.g., from video clips with static backgrounds and moving objects—would stress-test generalization. Extending the method to handle explicit object rotations would provide the degrees of freedom needed for interior-design scenarios.

REFERENCES

- [1] Omri Avrahami et al. “DiffUHaul: A Training-Free Method for Object Dragging in Images”. In: *SIGGRAPH Asia 2024 Conference Papers*. SA ’24. ACM, Dec. 2024, pp. 1–12. DOI: 10.1145/3680528.3687590. URL: <http://dx.doi.org/10.1145/3680528.3687590>.
- [2] Hanshen Zhu et al. *Training-free Geometric Image Editing on Diffusion Models*. 2025. arXiv: 2507.23300 [cs.CV]. URL: <https://arxiv.org/abs/2507.23300>.
- [3] Jiawei Ren et al. *Move Anything with Layered Scene Diffusion*. 2024. arXiv: 2404.07178 [cs.CV]. URL: <https://arxiv.org/abs/2404.07178>.
- [4] Karran Pandey et al. *Diffusion Handles: Enabling 3D Edits for Diffusion Models by Lifting Activations to 3D*. 2023. arXiv: 2312.02190 [cs.CV]. URL: <https://arxiv.org/abs/2312.02190>.
- [5] R. Venkatesh Babu Rishubh Parihar Sachidanand VS. “Zero-Shot Depth-Aware Image Editing with Diffusion Models”. In: *International Conference on Computer Vision (ICCV)*. 2025.
- [6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: <https://arxiv.org/abs/2006.11239>.
- [7] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG]. URL: <https://arxiv.org/abs/2010.02502>.
- [8] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV]. URL: <https://arxiv.org/abs/2112.10752>.
- [9] Ron Mokady et al. *Null-text Inversion for Editing Real Images using Guided Diffusion Models*. 2022. arXiv: 2211.09794 [cs.CV]. URL: <https://arxiv.org/abs/2211.09794>.
- [10] Lihe Yang et al. *Depth Anything V2*. 2024. arXiv: 2406.09414 [cs.CV]. URL: <https://arxiv.org/abs/2406.09414>.
- [11] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: <https://arxiv.org/abs/2304.02643>.
- [12] Xi Chen et al. *AnyDoor: Zero-shot Object-level Image Customization*. 2024. arXiv: 2307.09481 [cs.CV]. URL: <https://arxiv.org/abs/2307.09481>.
- [13] Its Ahmad. *Indoor Scenes CVPR 2019*. Kaggle. 2019. URL: <https://www.kaggle.com/datasets/itsahmad/indoor-scenes-cvpr-2019/data>.
- [14] Yikai Wang et al. *Towards Enhanced Image Inpainting: Mitigating Unwanted Object Insertion and Preserving Color Consistency*. 2025. arXiv: 2312.04831 [cs.CV]. URL: <https://arxiv.org/abs/2312.04831>.

APPENDIX A ADDITIONAL VISUALIZATIONS



Fig. 12: Results from no-scaling, disparity, and absolute depth configurations with FID, SSIM and MaskIoU metrics.