SeeThrough3D: Occlusion Aware 3D Control in Text-to-Image Generation

Supplementary Material

A. Overview

This appendix provides additional analysis, details about the dataset and model implementation, experimental discussion referenced in the main paper and extended qualitative results and comparisons. To skim over this material, the reader is advised to go through the figures and captions, which have been endowed with sufficient detail to understand the key content.

B. Dataset

B.1. The rendering pipeline

We collect 39 assets from Objaverse [15] and SketchFab [59] repositories from the internet. However, these assets are not aligned, making it difficult to define a canonical orientation for the objects. Hence, we align these assets manually in Blender [12] to ensure that their canonical front directions are aligned with the +Y axis. We further scale each asset to match relative real-world dimensions, for example, the size of jeep is smaller than an elephant, the scale values were obtained using Gemini 2.5 Pro [11]. We place the aligned assets in a Blender environment (upto 4 objects per scene) and add a virtual camera to render the scene from a given viewpoint. Specifically, we define a hemispherical region around the origin of a fixed radius R, within which all objects are placed. The camera lies at the surface of the hemisphere, always pointing towards the origin.

However, randomly placing the assets and the camera might result in unnatural-looking compositions, such as those where objects are colliding with each other. Additionally, as described in the ablations section (main paper), a key requirement is that the objects must be heavily occluded to ensure optimal training of our model. To cater to the above requirements, we adopt a procedural generation, where the scene configuration (camera and object placements) is first randomly sampled from a uniform distribution of parameters with some predefined constraints. This is followed by a filtering logic to remove the poor-quality examples.

Filtering based on occlusion: We filter the rendered scenes according to the extent of occlusion, to ensure heavy occlusion scenarios. For this, we require a metric to measure the extent to which an object is occluded. Therefore, we define a visibility ratio x, which is the ratio of visible area $\mathbf{v}$ of the object to the total area $\mathbf{a}$ of the object. The values $\mathbf{v}$ and $\mathbf{a}$ are measured using object segmentation masks, obtained through Blender [12]. We filter out cases where $x > 0.7$ for all objects in the scene, i.e., no object is occluded *enough*. Similarly, we filter out cases where $x < 0.3$ for any object, to ensure that each object is adequately visible in the image.

Filtering based on object size: We filter out cases where an object is too small or too large, to avoid unnatural-looking images. We filter based on the largest side of 2D object bounding boxes in the renderings. Specifically, we ensure that the largest side of the 2D bounding box must be within 0.125 and 0.750 of the image size.

B.2. Augmentations

Training solely on these rendered images from Blender risks overfitting to synthetic backgrounds [9, 49], due to limited realism and lack of diversity in object appearance and backgrounds. Since creating highly varied 3D scenes is an expensive process, we adopt a scalable alternative. We generate realistic augmentations for the rendered images that follow the same layout but are rich in terms of appearance diversity. For each rendered image, we extract its depth and pass it through a depth-to-image generation pipeline (FLUX.1-Depth-dev) [35] to synthesize realistic images that preserve the same spatial layout.

Filtering augmentation samples. Although this pipeline produces high-quality results, it occasionally misaligns objects with their intended depth regions, causing incorrect placements. For instance, on the left pane in Fig. 12(a), the depth-to-image model incorrectly generates a pigeon, instead of a crow, according to the original layout. We mitigate such issues by applying object-level CLIP-based filtering [56] to retain only those augmentations that adhere to the original layout. For this, we first obtain the object segmentation masks using Blender [12], and use these to obtain cropped object segments in the augmented image (see Fig. 12(b)). Next, we compute CLIP similarity between these object segments and corresponding text description (*e.g.* cat, pigeon, etc.), as shown in Fig. 12(c). If any object has a CLIP score less than the threshold value of 0.25, the augmented image is filtered out. High CLIP scores for all object segments (as shown on the right pane in Fig. 12) indicate accurate layout adherence, and these images are included in the training dataset. We visualize some examples from our training dataset in Fig. 15.

B.3. Statistics

We present various statistics of our training dataset in Fig. 13. (a) We plot the distribution ofthe minimum visibility ratio for any object in the scene. Since our filtering strategy favors heavy occlusion scenarios, we observe that there is a bias towards low visibility ratio cases. (b) Next, we observe that the distribution of orientation val-

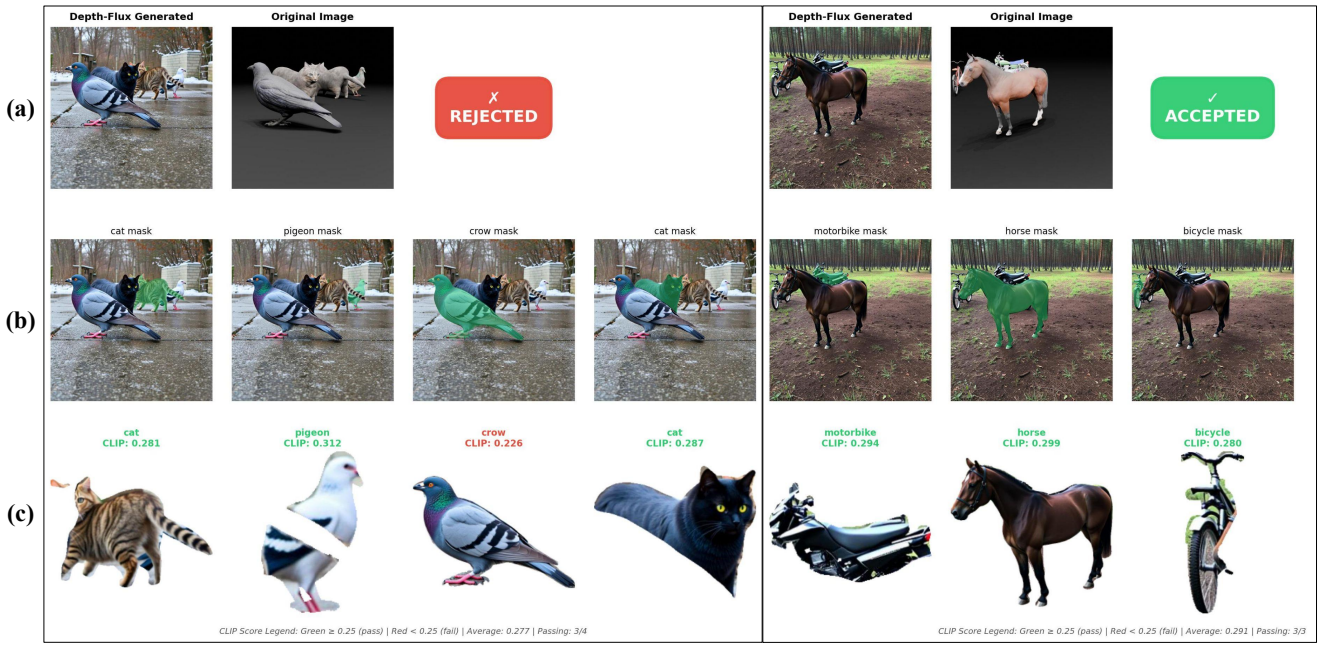


Figure 12. **CLIP filtering on augmentations:** We use a depth-to-image model (FLUX.1-Depth-dev [35]) to generate realistic augmentations of the rendered images (a). However, the depth-to-image model occasionally misaligns objects with their intended depth regions, causing incorrect placements. For example, on the left pane, the depth-to-image model incorrectly generates a pigeon in place of a crow according to original layout. We mitigate such issues by applying object-level CLIP-based filtering [56] to retain only those augmentations that adhere to the original layout. For this, we first obtain the object segmentation masks using Blender [12], and use these to obtain cropped object segments in the augmented image (b). Next, we compute CLIP similarity between these object segments and corresponding text description (*e.g.* cat, pigeon, etc.), as shown in (c). If any object has a CLIP score less than the threshold value of 0.25, the augmentation is filtered out. High CLIP scores for all object segments (as shown on the right pane) indicates accurate layout adherence, and these images are included in the training dataset.

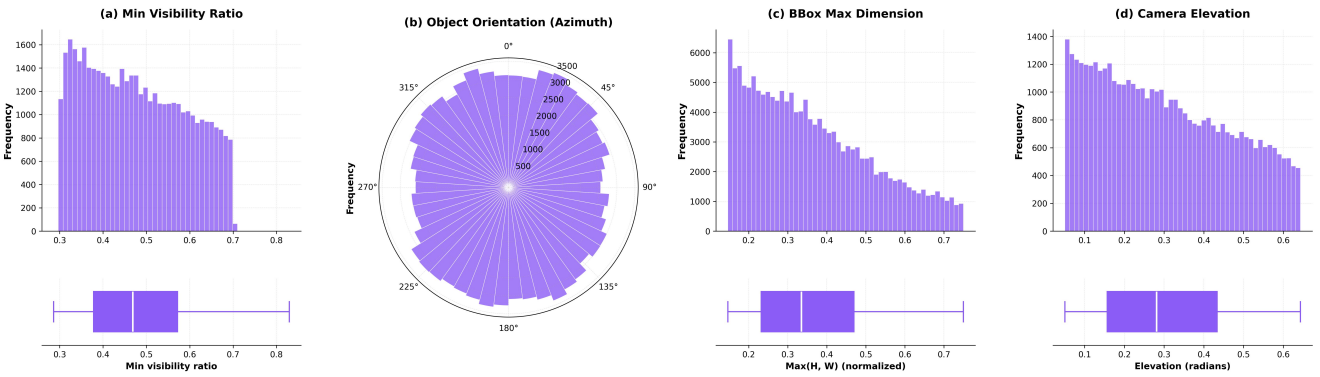


Figure 13. **Statistics of training dataset:** (a) We plot the distribution of minimum visibility ratio for any object in the scene. Since our filtering strategy favors heavy occlusion scenarios, we observe that there is a bias towards cases with low visibility ratios. (b) Next, we observe that the distribution of orientation values is roughly uniform, thus avoiding any unwanted biases. (c) Interestingly, we observe that the frequency of examples with large 2D bounding box dimension shows a decreasing trend. This is because smaller object sizes enable placement of multiple objects in a scene, while ensuring all of them are visible. (d) High camera shots tend to have weaker occlusions compared to low camera shots; for instance, there are very little inter-object occlusions in bird's-eye-view of a scene (high camera elevation). Since our data selection process favors high occlusion scenarios, renders with low camera are usually favored by the rendering pipeline algorithm, explaining the observed trend.

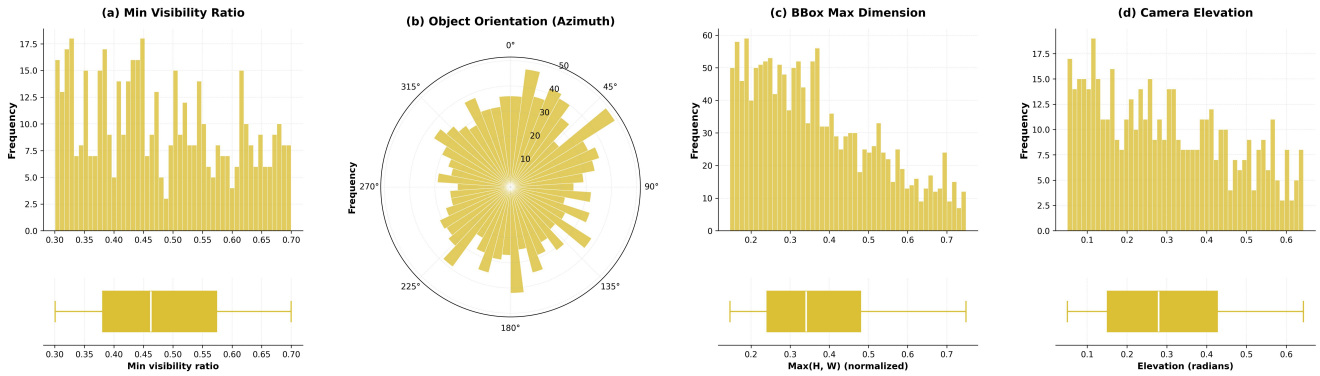


Figure 14. **Statistics of *3DOcBench* evaluation benchmark:** Similar to the training dataset, we observe that the 3DOcBench evaluation benchmark contains (a) heavy occlusion scenarios, (b) roughly uniform distribution of orientations, (c) higher frequency of smaller objects, measured using 2D bounding box dimension, and (d) large number of cases with low camera elevation and consequently high inter-object occlusion.

ues is roughly uniform, thus avoiding any unwanted biases. (c) Interestingly, we observe that the frequency of examples with large 2D bounding dimensions shows a decreasing trend. This is because smaller object sizes enable the placement of multiple objects in a scene, while ensuring all of them are visible. (d) By common observation, high camera shots tend to have weaker occlusions compared to low camera shots. For instance, there are few occlusion scenarios in bird's-eye-view (high camera elevation) of a scene. Since our data selection process favors high occlusion scenarios, renders with low camera are favored by the rendering pipeline algorithm, explaining the observed decreasing trend.

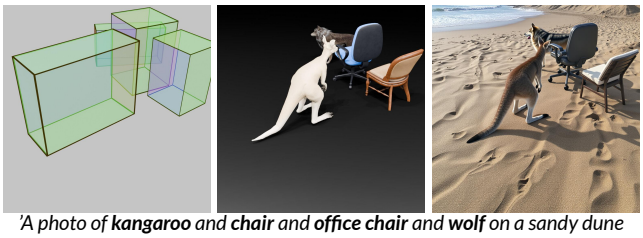
C. *3DOcBench* benchmark details

For constructing our evaluation benchmark, *3DOcBench*, we use the same procedural generation to prepare the training dataset (see Sec. B.1). Specifically, we construct scene layouts in Blender [12] with the 3D assets and camera placed in random locations, and filter the layouts based on whether they meet the constraints of occlusion and object size (see Sec. B.1). We present various statistics of the benchmark in Fig. 14. Similar to the training dataset, we observe that the 3DOcBench evaluation benchmark contains (a) heavy occlusion scenarios, (b) roughly uniform distribution of orientations, (c) higher frequency of smaller objects, measured using 2D bounding box dimension, and (d) a large number of cases with low camera elevation and consequently high inter-object occlusion.

D. Overlapping regions

In the proposed object binding strategy, the OSCR tokens at the intersection of two rendered bounding boxes attend to all participating object tokens in the text prompt. However, at first glance, it seems that attending to multiple object semantics would lead to semantic bleeding and visual artifacts at object boundaries. Upon investigation, however, we found that the generated images feature sharp occlusion boundaries without object attribute mixing. To understand this, we visualize the attention maps between image and text tokens, and find that object features are segregated in the model's latent space. We analyse the attention maps through 8 complex scene layouts (of two objects) with heavy occlusion scenarios in Fig. Fig. 16. As we can see, the attention maps clearly distinguish between the foreground and background objects with appropriate occlusions. This indicates the inherent model's capability in handling object occlusions, and our method provides a new interface to accurately generate such scenes, which is challenging to do with text alone.

Selecting layers for attention visualization. We performed a simple analysis for choosing the appropriate layers for attention visualization. We use Segment Anything [30] on the generated images followed by manual filtering to segment out individual object regions. Finally, we measure spatial alignment between image to object token attention and ground truth segmentation using correlation coefficient (CC). We analyze CC values across space (DiT layers) and time (denoising timesteps), results are visualized in Fig. 17. The results highlight that spatial alignment is high for very specific layers in the DiT, particularly for layers 11 to 23. Also, spatial alignment tends to emerge at around $5th$ denoising timestep (out of 25 timesteps). We use the resulting CC matrix to filter top-50 (layer, timestep) combinations which show highest spatial alignment, and average image to text attention for each object. The results are visualized in Fig. 16.



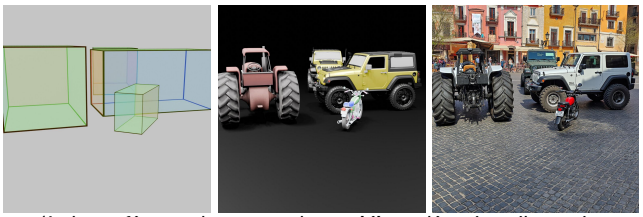
*'A photo of **kangaroo** and **chair** and **office chair** and **wolf** on a sandy dune near the ocean with gentle waves.'*

*'A photo of **elephant** and **jeep** and **bulldozer** in a wide open desert with distant mesas and clear air.'*

*'A photo of **wolf** and **chair** and **pig** and **dog** amongst blooming lavender fields under a sunny sky.'*

*'A photo of **teddy** and **pigeon** and **teddy** and **teddy** on a kitchen counter.'*

*'A photo of **cat** and **crow** and **teddy** and **pigeon** on a stack of old books in a library.'*



*'A photo of **jeep** and **tractor** and **motorbike** and **jeep** in a vibrant city square with historic architecture.'*

Figure 15. **Samples from our training dataset:** We create scenes in Blender [12] by placing 3D assets in controlled configurations and defining the rendering camera viewpoint. The object arrangements and camera viewpoint are controlled to ensure strong inter-object occlusions, while ensuring that each object is sufficiently visible in the image. Along with the main image, we render the corresponding OSCR representation, which consists of color-coded translucent 3D bounding boxes of the objects. The rendered images are further augmented using a depth-to-image pipeline to obtain realistic images that follow the same layout as shown in Fig:12.

E. Implementation details

We build upon FLUX.1-dev [35] as our base model. We patch it with 128-rank LoRA adapters, applied on query, key and value projections in every attention layer. Additionally, we set the LoRA scale to 0 for the text and image tokens to preserve the strong text-to-image prior of the base model [61, 62, 79]. We train our model with a learning rate of 10^{-4} using the AdamW optimizer for $30K$ steps with an effective batch size of 2. The first $25K$ training steps use an image resolution of 512, followed by a resolution of 1024 for the next $5K$ steps. We found that such staged training helps improve realism in the generated images. The complete training takes around 9 hours on $2X$ NVIDIA $H100$ GPUs (one image per GPU). Our implementation is based on PyTorch [52] and Hugging Face Diffusers [63] framework.

To enable personalization, we introduce an additional 'subject' LoRA of the same rank (128) on the reference image tokens. Both the LoRA's are finetuned on our personalization dataset (see Sec. I) for $7.5K$ iterations.

F. Taking control with SeeThrough3D

OSCR representation encodes various 3D attributes of a scene, such as object orientation, size and location, along with camera viewpoint as well as occluded object regions. This enables SeeThrough3D to control all the properties in jointly. Additionally, since our method preserves the strong prior of the base text-to-image model, it can generate diverse visual appearance of both objects and background, solely through text prompt control. These diverse forms of control offered by SeeThrough3D are summarized in Figs. 18 and 19. Note that **all the images in these figures are generated using the same random seed**, highlighting the effectiveness of control. Notably, the model is able to preserve occlusion consistency even despite heavy overlaps, such as low camera elevation (d1, Figs. 18 and 19), (b4, Fig. 18). These results indicate the preciseness of control enabled by our method, enabling various applications in design and architecture.

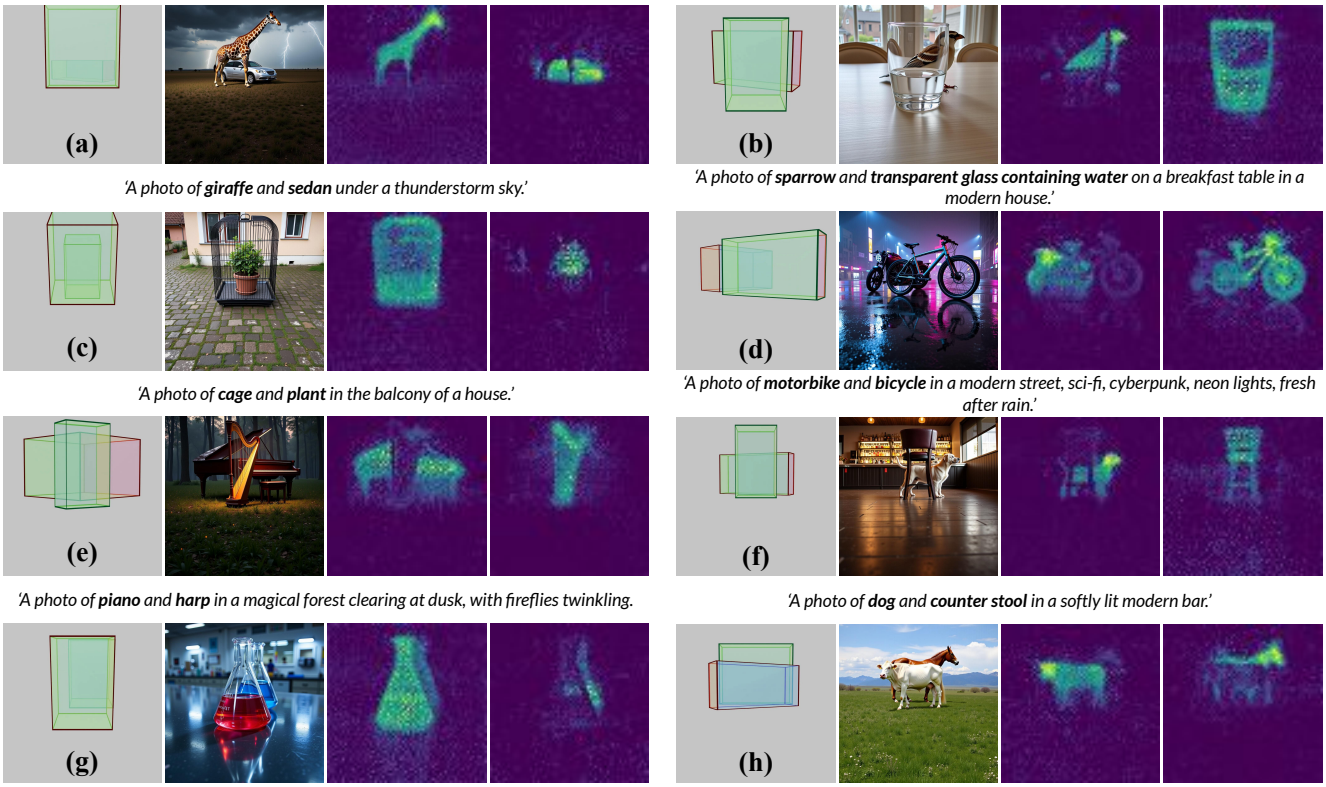


Figure 16. **Visualizing object disentanglement in latent space:** We use the layouts (shown in first frame) to condition our model, the outputs are shown in second frame. We store the intermediate attention maps from image tokens to object tokens in the text prompt, visualized in third and fourth images. Evidently, the attention maps reveal occlusion boundaries, and show some interesting patterns. Notably, in cases involving transparent objects like water (b) and chemical flask (g), the physically hidden regions of sparrow and flask respectively are visible in attention map. Even in case of semantically similar categories, such as cow and horse (a), flasks with differently colored chemicals (g), motorbike and bicycle (d) the attention is highly localized, with only minimal leakage.

G. Additional baseline comparisons

In the main paper, we have compared against **3D scene control** methods, *LooseControl* [4] and *Build-A-Scene* [19]. These methods are directly relevant to ours, since they enable control over 3D scene layout, including object placement, orientation and camera viewpoint. Here we compare against baselines which specifically allow control over 3D object orientation only, without controlling 3D object placement or camera viewpoint. We compare against two baselines, *ORIGEN* [43] and *Compass Control* [49]. ORIGEN enables control over object orientation using a one step generative model. Specifically, they perform initial noise optimization according to a reward function which penalizes the mismatch between orientation of the generated object and the input orientation angle. The generated object orientation that is measured using Orient Anything [67]. However, they do not provide control over locations of the objects. Compass Control, on the other hand, enables control over object orientation along with 2D object layouts. They learn an adapter which maps object orientation to a per object *compass* embedding. These embeddings are then used to condition the generative process through cross attention. The cross attention maps of compass and object tokens in prompt are constrained to respective 2D bounding boxes to enable disentangled orientation control for multi-object scenes.

Analysis: We present comparison results against these baselines in Fig. 20 and Tab. 3. Since ORIGEN [43] does not allow 2D layout control, it is not compatible with our quantitative evaluation that focuses on layout adherence (see **Evaluation metrics** in the main paper), and we only present qualitative comparisons. We observe that Compass Control is not able able to handle complex occlusions ($A1-4$), and mixes object attributes in case of heavy overlaps ($A5-6$), resulting in low objectness score. ORIGEN fails to generate some objects in the scene (B1-6). Additionally, its outputs contain artifacts arising from poor noise optimization (B2). Additionally, ORIGEN is limited to one-

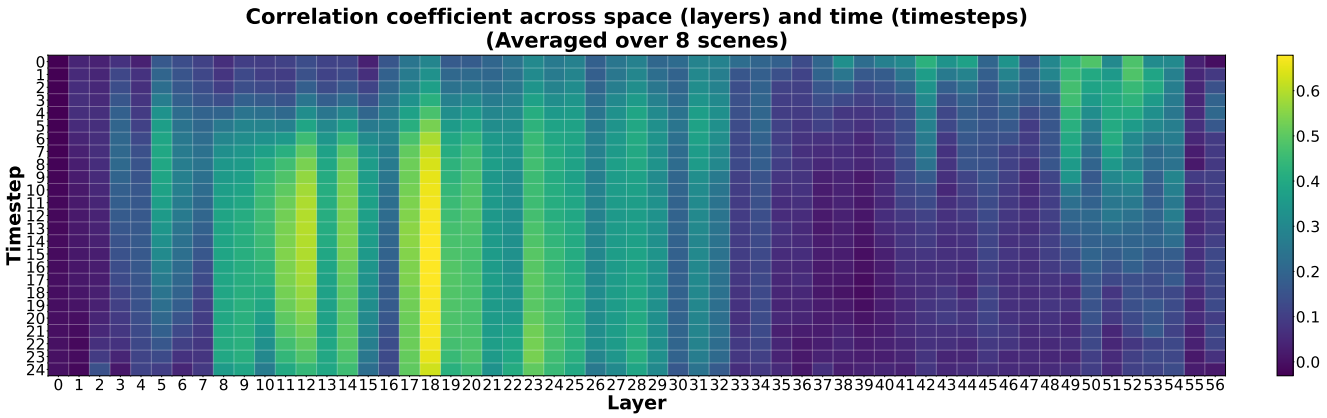


Figure 17. **Measuring spatial alignment of image to object attention using correlation coefficient (CC):** We create a dataset of 8 complex layouts containing strong occlusion scenarios (see Fig. 16). We obtain SeeThrough3D's outputs on these layouts, and store intermediate attention maps from image tokens to object tokens in the text prompt. Next, we run Segment Anything [30] on the generated outputs to obtain object-level segmentation masks. Finally, we use correlation coefficient (CC) to measure alignment between the ground truth object segment and corresponding image to object attention maps. We compute the CC across space (layers) and time (denoising timesteps), and the obtained heatmap reveals interesting insights. For a given layer, timestep combination, a high CC value indicates strong spatial awareness. We observe that very specific layers in the DiT are spatially aware; early layers from 8 to 25, after which the spatial awareness decreases sharply. Secondly, the spatial properties in attention emerge after 5th denoising step (out of 25 steps) in these layers. The pattern of spatially aware layers is very irregular, indicating that different layers in the DiT contribute very differently to the generated image, consistent with findings from [1].

step generative models, and hence suffers from low image fidelity. In contrast, our method is able to model complex occlusions (E1-6) without attribute mixing, indicating its effectiveness.

H. More on angular error evaluation

Two of our baselines, *LooseControl* [4] and *Build-A-Scene* [19] use layout depth maps as condition for text-to-image model. While providing 3D placement cues, the layout depth representation fails to capture precise 3D orientation, leading to poor orientation accuracies, as indicated by high angular error values in Tab. 3. Specifically, we observe a large number of 180° flips, because bounding box depth does not encode the front-facing direction of the object. Therefore, we evaluate a *relaxed* angular error which does not penalize 180° flips in the generated objects, results tabulated in Tab. 3. We observe that 3D layout control baselines, LooseControl and Build-A-Scene show slightly improved results compared to *LaRender* [76] and *VODiff* [37], which are not orientation aware. *Compass Control* [49] encodes orientation value through an adapter, hence performs better than the other baselines on angular error. In contrast, our OSCR representation explicitly encodes orientation in the image space using color-coding, thus enabling precise orientation control, performing favorably compared to all existing methods.

Baselines	depth ord.↑	obj. score↑	angular err.↓	text align.↑	KID($\times 10^{-3}$)↓	180° flip ang. err.↓
VODiff [37]	0.68	19.70	92.73	29.51	15.40	41.38
LooseControl [4]	0.82	20.02	89.88	28.43	14.32	37.48
Build-A-Scene [19]	0.89	21.0	91.62	28.05	20.12	32.23
LaRender [76]	1.02	21.83	89.63	30.20	13.46	41.19
CompassControl [49]	0.87	20.60	66.29	29.76	13.01	35.79
Ours	**1.46**	**22.86**	**47.92**	**31.87**	**5.43**	**25.72**

Table 3. **Quantitative comparison:** In the main paper, we did not compare against methods that only enable partial 3D control: *Compass Control* [49] and *ORIGEN* [43]. These baselines do not allow for 3D layout control, and primarily focus on object orientation control. While Compass Control allows for 2D layout control, ORIGEN does not, and hence it is not compatible with our quantitative evaluation (see **Evaluation metrics** in the main paper). Results for Compass Control are presented in yellow. It implicitly encodes orientation using an adapter, hence performs better than the other baselines in angular error. Further, we evaluate a *relaxed* angular error, which does not penalize 180° flips in the generated object (violet column). This caters to layout depth based methods *LooseControl* [4] and *Build-A-Scene* [19], which do not encode a front-facing direction for the objects, thus result in such 180° flips. Our OSCR representation explicitly encodes orientation in the image space using color-coding, thus enabling precise orientation control, outperforming all baselines.

I. Personalization

We show that SeeThrough3D can be finetuned for occlusion-aware 3D control of personalized objects (see Fig. 21). This is achieved by learning a separate 'subject' LoRA to fuse appearance attributes from personalized object image into the generation process, building upon prior work on conditioning diffusion transform-

Figure 18. **Taking control with SeeThrough3D:** We demonstrate the individual controls that our approach offers over the scene composition, which includes 3D attributes such as (a) object orientation, (b) object size, (c) object location, (d) scene camera elevation, as well as (e) text prompt and object semantics, all while ensuring occlusion consistency. Notably, all the images in this figure were generated **using the same random seed**, highlighting the effectiveness of control. **Disentangled control:** In (a), (b) and (c), we are able to control the 3D attributes of one object (Rolls Royce), without altering the other object, indicating disentangled control. Notice how occlusion consistency is preserved even in case of heavy overlap (b4), when the white car has become very big, and in (d1), where the camera elevation is very low. The model is able to model interesting controls such as levitating objects (c4). Despite heavy overlaps, the **object attributes ('white Rolls Royce', 'yellow Ferrari') remain correctly bound** to respective objects without attribute mixing, highlighting effectiveness of our binding mechanism.

Figure 19. **Taking control with SeeThrough3D:** We demonstrate the individual controls that our approach offers over the scene composition, which includes 3D attributes such as (a) object orientation, (b) object size, (c) object location, (d) scene camera elevation, as well as (e) text prompt and object semantics, all while ensuring occlusion consistency. Notably, all images in this figure were generated **using the same random seed**, highlighting the effectiveness of control. **Disentangled control:** In (a), (b) and (c), we are able to control the 3D attributes of one object (red boat), without altering the other object, indicating disentangled control. Notice how occlusion consistency is preserved in challenging cases like (d1), where the camera elevation is very low. Despite heavy overlaps, the **object attributes ('green boat', 'red boat') remain correctly bound** to respective objects without attribute mixing, highlighting effectiveness of our binding mechanism.

Figure 20. **Qualitative comparisons with additional baselines:** We present comparisons with baselines *Compass Control* [49] and *ORIGEN* [43]. We observe that Compass Control is not able able to handle complex occlusions (A1-4), and mixes object attributes in case of heavy overlaps (A5-6). ORIGEN fails to generate some objects in the scene (B1-6), and its outputs contain visual artifacts arising from poor noise optimization (B2). Additionally, ORIGEN is limited to one-step generative models, and hence suffers from low image fidelity. In contrast, our method is able to model complex occlusions (E1-6) without attribute mixing, indicating its effectiveness.

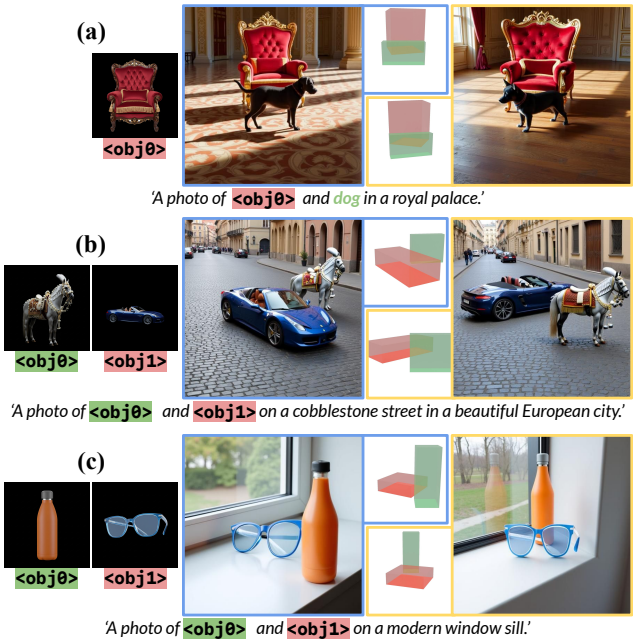


Figure 21. **Personalization:** We show that SeeThrough3D can be finetuned for occlusion-aware 3D control of personalized objects. This is achieved by learning a separate 'subject' LoRA to fuse appearance attributes from personalized object image into the generation process, building upon prior work on conditioning diffusion transformers [61, 62, 79]. This approach achieves such personalized 3D control **without need for any test-time tuning**. As shown in (a), we can **compose objects from multiple modalities**, such as dog (text) and royal chair (image). Interestingly, our model can **personalize object categories not seen during** training, such as bottle and glasses in (c), indicating strong generalization.

ers [79]. This approach achieves personalized 3D control **without need for any test-time tuning**. As shown in (a), we can **compose objects from multiple modalities**, such as dog (text) and royal chair (image). Interestingly, our model can **personalize object categories not seen during** training, such as bottle and glasses in (c), indicating strong generalization.

To train the personalization model, we suitably adapt our dataset for this task. Given a rendered image, we randomly choose an object and apply a texture to it in Blender [12] (see Fig. 22(a,b)). For this, we generate a small set of textures using FLUX [35] by prompting it to ensure high frequency details such as text and sharp patterns, some samples are shown in Fig. 23. We separately render the textured 3D asset, and use it as the reference image condition (see Fig. 22(b)). The orientation of the reference object is slightly altered to enable the model to reason about its 3D placement, and not just copy pixels from reference image. Finally, the textured object is placed back into the original image, and used as ground truth target for training the model (see Fig. 22(c)), conditioned on the reference image.

J. User study

We conducted an A/B user study where 60 participants were asked to choose between output of our method and a randomly chosen baseline. We evaluate a) image realism, b) layout adherence, and c) text prompt alignment. Results highlight high preference for our method in all evaluation categories (see Fig. 25).

K. Additional qualitative comparisons

We present additional qualitative comparisons with the main baselines in Figs. 27 and 28. Each example has been analyzed with reference to layout adherence and occlusion consistency (red text above each example). Results indicate that SeeThrough3D generates realistic images following precise 3D layouts while maintaining occlusion consistency, and outperforms all baselines.

(a) original rendered image from training data. (b) texture applied to object, used as reference image. (c) textured object placed back, used as target image.

Figure 22. **Adapting the training dataset for personalization:** (a) given a rendered image from the training dataset, we randomly choose an object and apply a texture to it in Blender [12] (see Fig. 23 for examples of generated textures). (b) We separately render the textured 3D asset, and use it as reference object condition. (c) Finally, the textured object is placed back into the original image, and used as ground truth target for training the model.

Figure 23. **Examples of generated textures:** We generate textures using FLUX [35], for preparing data for the personalization task. Notice how the textures contain high frequency features, induced by text and sharp patterns.

*'A photo of **parrot** behind a **cage** in a forest environment.'*

Figure 24. **Limitations:** (a) Our model is built upon FLUX [35] which fails to generate some out of distribution cases, such as a parrot outside a cage. Consequently, our model also struggles to generate such cases.

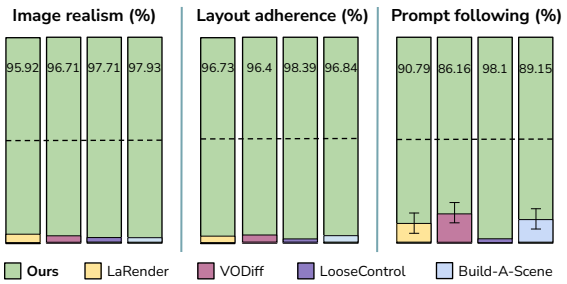


Figure 25. **User study:** Each bar indicates the % of times our method's output was preferred over the baseline, for each category.

L. Additional qualitative results

We present additional results of our method in Fig. 29. For each example, we have shown the OSCR layout alongside the generated image; the correspondence from boxes to individual objects has been omitted here for clarity.

M. User interface

One of the motives of SeeThrough3D is to enable creative artists to precisely control various 3D elements of a generated image, such as scene layout and camera viewpoint. To ease the design process, we built an intuitive web interface, which allows the user to construct 3D layouts and control camera viewpoint. The interface allows the user to add boxes for various objects, edit their dimensions, 3D placement, and specify a text description for each object. The interface also comes with pre-defined template dimensions of common objects such cars, animals, etc. which can be used. A screenshot of the interface can be seen in Fig. 26.

N. Limitations

Since our method conditions a pretrained text-to-image model (FLUX [35]), it is limited by the capabilities of the base model. For instance, FLUX sometimes fails to generate out of distribution cases, such as a parrot behind a birdcage, with realistic occlusion. Consequently, our method, which is built upon the prior of FLUX, also fails to generate such cases, as shown in Fig. 24. Additionally, personalization requires that all the reference image tokens be present in the transformer's context; this leads to higher VRAM requirements, especially for multi-subject personalization.

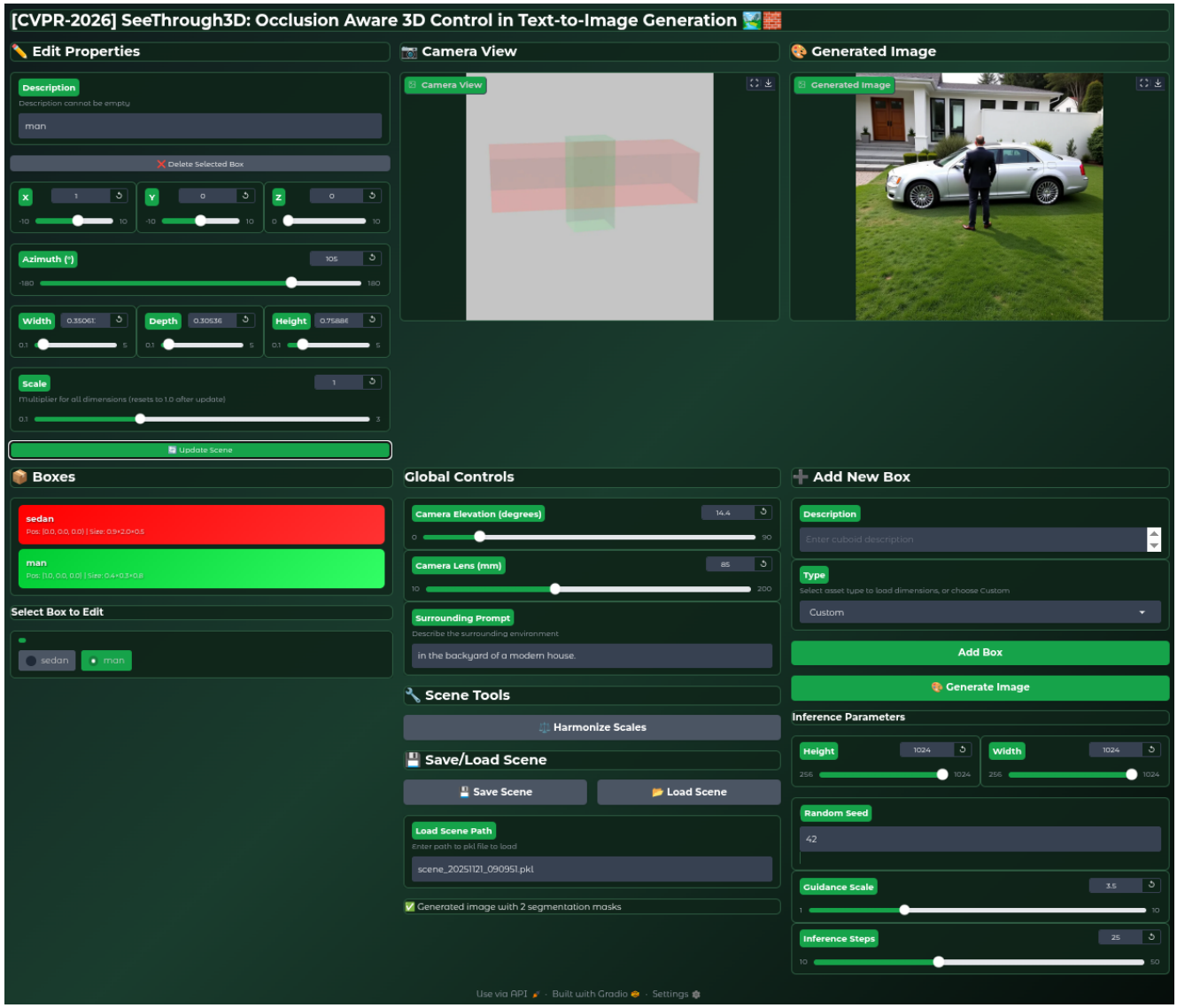
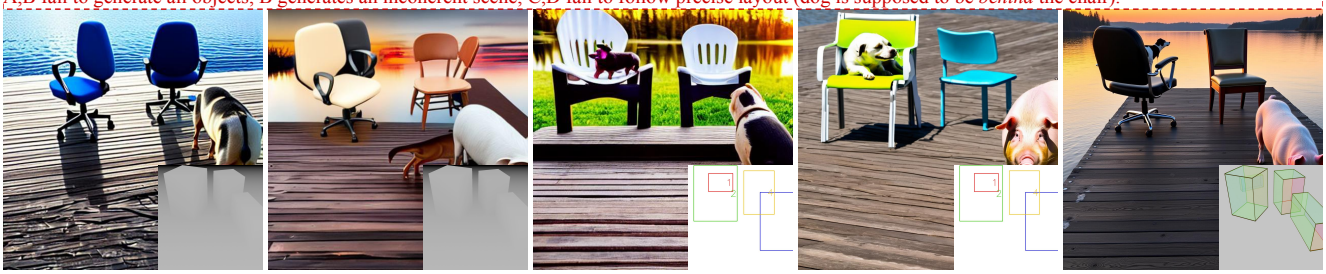


Figure 26. We built an intuitive UI to enable the user to easily design layouts for using our model. The UI enables the user to add objects, edit their placement, orientation and dimensions, and provide a text description for each object. Additionally, it allows the user to set the camera parameters.

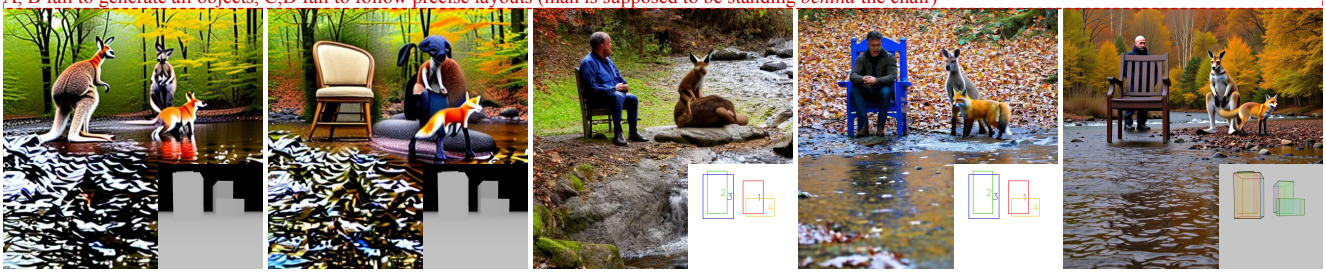
A,B fail to generate all objects, B generates an incoherent scene, C,D fail to follow precise layout (dog is supposed to be *behind* the chair).



(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of dog and office chair and pig and chair on a wooden pier overlooking a serene lake at sunset.'

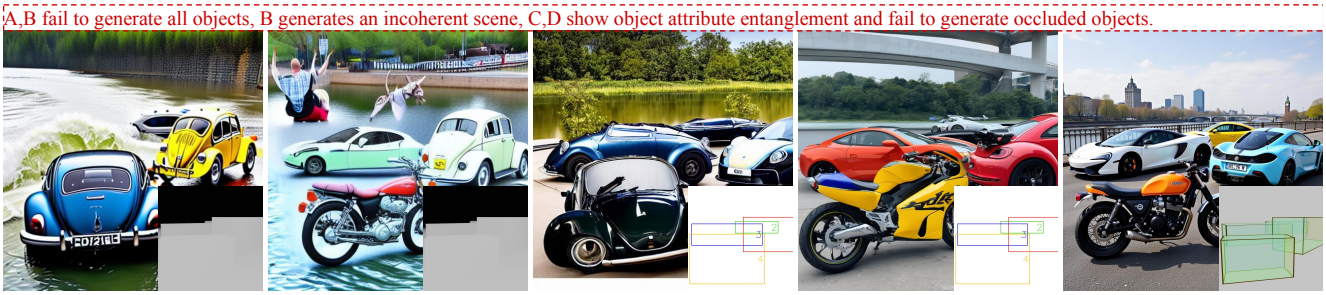
A, B fail to generate all objects, C,D fail to follow precise layouts (man is supposed to be standing *behind* the chair)



(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of kangaroo and man and chair and fox next to a babbling brook in an autumnal woodland.'

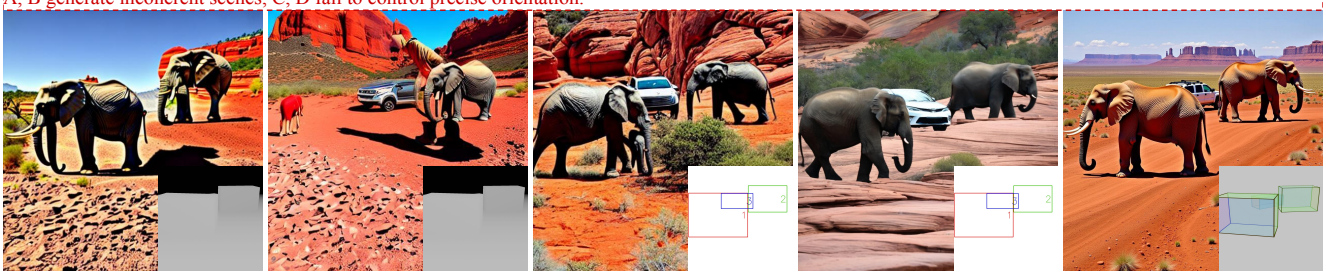
A,B fail to generate all objects, B generates an incoherent scene, C,D show object attribute entanglement and fail to generate occluded objects.



(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of vw beetle and mclaren and mclaren and motorbike on a bridge spanning a wide river in a metropolitan area.'

A, B generate incoherent scenes, C, D fail to control precise orientation.

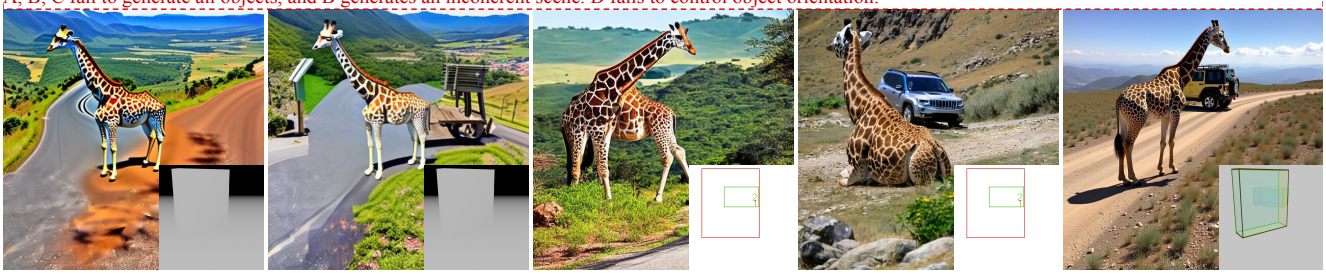


(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of elephant and elephant and suv in an expansive national park with majestic red rock formations.'

Figure 27. We present qualitative comparisons with the baselines from the main paper, which are categorized into **3D layout control:** *LooseControl* [4] and Build-A-Scene [19]; **Occlusion control:** *VODiff* [37] and LaRender [76]. Each example has been analyzed with reference to layout adherence, occlusion consistency and realism (**red text** above each example).

A, B, C fail to generate all objects, and B generates an incoherent scene. D fails to control object orientation.



(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of giraffe and jeep in a mountainous region with winding roads and panoramic views.'

All baselines fail to generate a small, heavily occluded objects (the tiger). B shows visual artifacts due to inversion, and C has heavy attribute mixing.

(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of tiger and coupe and van and lamborghini in a vast, empty field during a foggy morning.'

All baselines fail to generate a small, heavily occluded object (the dog behind the man). B shows visual artifacts due to inversion.

(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

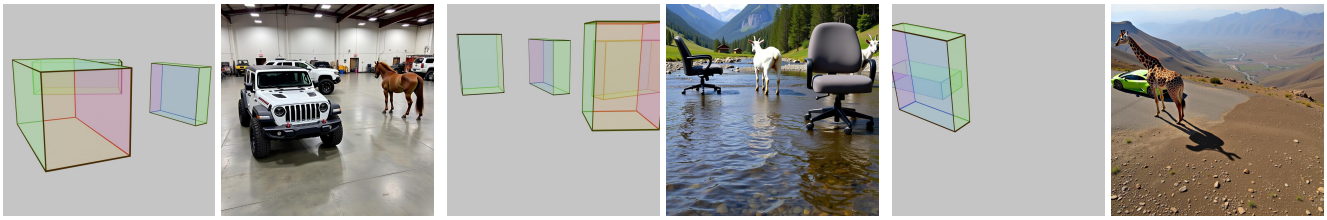
''A photo of wolf and dog and man by a calm river at dawn, reflecting the pastel sky.'

A, B, C generate incorrect layouts and fail to generate some objects with good fidelity. D mixes the objects van and table, leading to incorrect generation.

(**A**) LooseControl (**B**) Build-A-Scene (**C**) VODiff (**D**) LaRender (**E**) Ours

'A photo of lamborghini and table and van traversing a rocky riverbed in a deep canyon under a hot sun.'

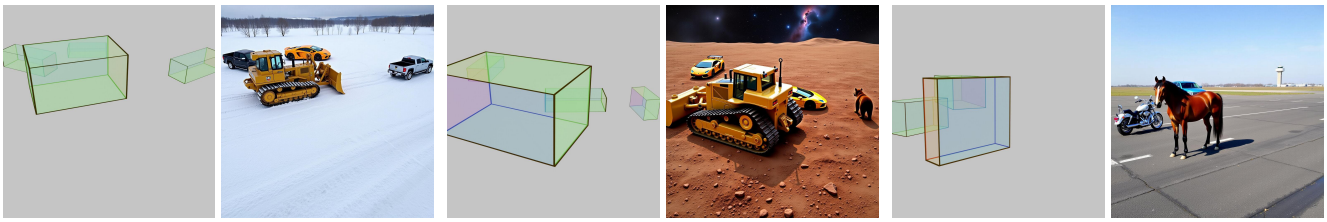
Figure 28. We present qualitative comparisons with the baselines from the main paper, which are categorized into **3D layout control:** *LooseControl* [4] and Build-A-Scene [19]; **Occlusion control:** *VODiff* [37] and LaRender [76]. Each example has been analyzed with reference to layout adherence, occlusion consistency and realism (**red text** above each example).



*'A photo of **suv** and **horse** and **jeep** in a large open garage with tools and other vehicles.'*

*'A photo of **goat** and **office chair** and **goat** and **office chair** on a rustic bridge over a clear mountain stream.'*

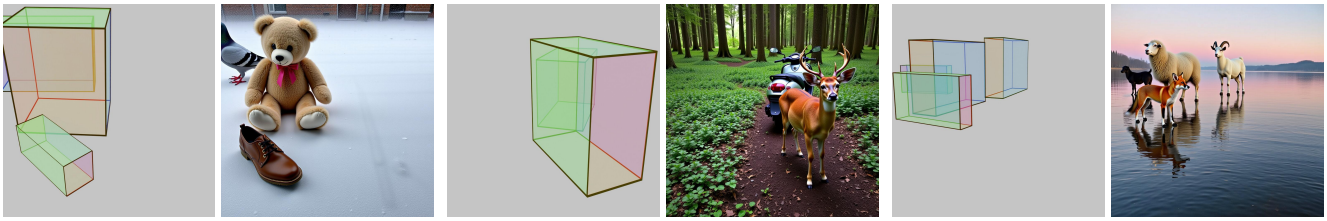
*'A photo of **lamborghini** and **giraffe** in a mountainous region with winding roads and panoramic views.'*



*'A photo of **lamborghini** and **pickup truck** and **pickup truck** and **bulldozer** on a snow-covered tundra with sparse trees and a frozen lake."*

*'A photo of **lamborghini** and **lamborghini** and **bear** and **bulldozer** on a desolate, rocky planet surface with a distant nebula.'*

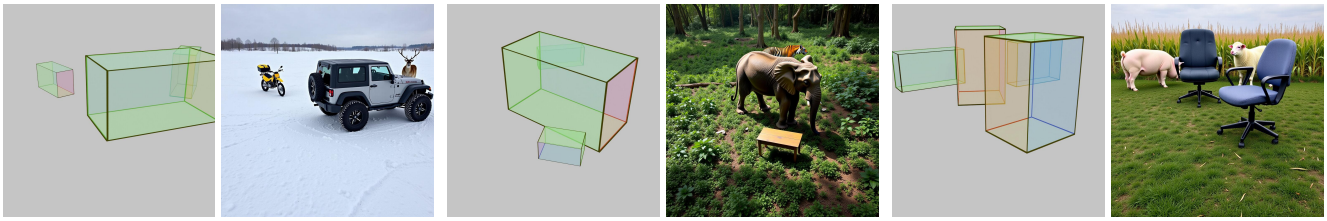
*'A photo of **pickup truck** and **motorcycle** and **horse** on an airstrip with a control tower in the background.'*



*'A photo of **pigeon** and **teddy** and **shoe** on a frosted windowpane, leaving tiny prints.'*

*'A photo of **scooter** and **deer** on a remote forest trail with ancient trees and thick undergrowth.'*

*'A photo of **dog** and **goat** and **sheep** and **fox** by a calm river at dawn, reflecting the pastel sky."*



*'A photo of **motorbike** and **deer** and **jeep** on a snow-covered tundra with sparse trees and a frozen lake.'*

*'A photo of **tiger** and **elephant** and **table** in a dense jungle with exotic foliage and filtering sunlight.'*

*'A photo of **sheep** and **pig** and **office chair** and **office chair** at the edge of a cornfield with stalks reaching high.'*

Figure 29. **Qualitative results:** We present additional results of our method. For each example, we have shown the OSCR layout alongside the generated image; the correspondence from boxes to individual objects has been omitted here for clarity.