A Supplemental Material

This supplementary material is organized as follows: Section A.1 provides more details about Semantics-Layout Variation AutoEncoder; Section A.2 introduces the Stable Diffusion and its attention mechanism; Section A.3 describes the implementation of the Multi-Layer Sampler in detail; Section A.4 covers more ablation studies; Section A.5 presents more qualitative results, including comparison visualization and graph manipulation; Section A.6 discusses the limitations of this study. Section A.7 delves into the broader societal impacts of this work. The core script is zipped and attached to the supplementary material.

A.1 Semantics-Layout Variation AutoEncoder

Recall that we apply the triplet-GCN-based CVAE architecture in Section 3.1. Each triplet-GCN layer in the encoder and decoder takes the node and edge embeddings. Specifically, the GCN_l mentioned in the paper uses two cascading MLPs $\{\mathrm{mlp}_1, \mathrm{mlp}_2\}$ to deal with node and edge embeddings:

$$(\psi_i^l, \phi_{ij}^{l+1}, \psi_j^l) = \mathrm{mlp}_1(\phi_i^l, \phi_{ij}^l, \phi_j^l), l \in \{0, \ldots, L-1\}, \tag{12}$$

$$\phi_i^{l+1} = \psi_i^l + \mathrm{mlp}_2(\mathrm{avg}(\psi_j^l \mid j \in \mathcal{N}_\mathcal{E}(o_i))), \tag{13}$$

where l denotes the layer index of encoder or decoder, $\mathcal{N}_\mathcal{E}$ denotes the neighbor index set for each node, avg denotes the average pooling operation, ϕ and ψ denote intermediate features. Hence, mlp_1 conducts message passing among interconnected nodes and updates the edge features, while mlp_2 aggregates features from all neighboring nodes and updates its features. For graph union encoder, we let $(\phi_i^0, \phi_{ij}^0, \phi_j^0) = (\mathcal{O}_i, \mathcal{E}_{ij}, \mathcal{O}_j)$. The last embedding ϕ_i^L is parameterized to the Gaussian distribution $Z \sim \mathcal{N}(\mu, \sigma)$, where $\mu, \sigma \in \mathbb{R}^{D_z}$ output by two additional MLPs and D_z denotes the dimensional of latent space for node embedding.

A.2 Diffusion with Compositional Masked Attention

Stable Diffusion [20] is one of most popular text-to-image model. As described in Section 2.1, Stable Diffusion uses a U-Net ϵ_θ composed of convolution and transformer to estimate noise. The transformer includes two attention mechanisms, namely Cross-Attention, and Self-Attention.

Cross-Attention Layer. Text prompts are mapped to sequence embeddings by CLIP text encoder and integrated into UNet via Cross-Attention to guide the de-noising trajectory:

$$Attention(Q_{visual}, K_{text}, V_{text}) = softmax(\frac{Q_{visual}K_{text}^T}{\sqrt{d}}) \cdot V_{text} \tag{14}$$

where Q_{visual} denotes the Query from the visual token of the UNet, K_{text} and V_{text} denotes Key and Value from text embeddings, all of which are projected by linear layers, d denotes the dimension of Q_{visual}, K_{text}, and V_{text}.

Self-Attention Layer. Self-Attention captures self-related information within visual tokens:

$$Attention(Q_{visual}, K_{visual}, V_{visual}) = softmax(\frac{Q_{visual}K_{visual}^T}{\sqrt{d}}) \cdot V_{visual} \tag{15}$$

where Q_{visual}, K_{visual}, and V_{visual} separately represent the Query, Key, and Value in self-attention layers, which are projected by linear layers. The self-attention mechanism isolates the information flow between specific tokens by multiplying a mask $\mathbf{M}$ to the $Q_{visual}K_{visual}^T$. Since $\mathbf{M}$ is applied before $softmax$, the value of the isolated position is set to negative infinity $-inf$.

Compositional Masked Attention Layer. Based on the attention mask $\mathbf{M}$ that depends on layout $\mathcal{B}$, the Compositional Masked Attention can be expressed as:

$$Attention(Q_{CMA}, K_{CMA}, V_{CMA}) = softmax(\frac{Q_{CMA}K_{CMA}^T \odot \mathbf{M}}{\sqrt{d}}) \cdot V_{CMA} \tag{16}$$

where Q_{CMA}, K_{CMA}, and V_{CMA} individually represent the Query, Key, and Value derived from $\mathcal{V} \otimes \hat{\mathcal{C}}$, achieved through linear layer projections. We insert our proposed Compositional Masked Attention (CMA) between self-attention and cross-attention layers.

A.3 Multi-Layer Sampler

Layered Scene Representation. We decompose a controllable scene containing N_o objects into N_o layers. Different from SceneDiffusion [25], our approach involves each layer incorporating not only

separate latent code z_i and spatial layout b_i, but also integrating the interactive semantics s_i produced by the SL-VAE. Here we convert the layout parameter b_i to two parts: (1) a fixed *object-centric* binary mask $m_i \in \{0,1\}^{c\times w\times h}$ to solely show the geometric property of the object, and (2) a two-element offset $p_i = \{\mu_i, \upsilon_i\}$ to solely indicate its spatial locations, with μ_i and υ_i defining the horizontal and vertical movement range. We sample Gaussian noise individually for the initial latent code of each layer, i.e., $\mathcal{Z} = \{z_i^{(T)} \sim \mathcal{N}(0,1)\}_{i=1}^{N_o}$. Then we utilize the layout-converted non-overlapping masks $\{l_i\}_{i=1}^{N_o}$ to derive the aggregated latent code $\boldsymbol{z}$ from various layers:

$$\boldsymbol{z}^{(t)} = \sum_{i=1}^{N_o} l_i \odot \overline{shift}(\boldsymbol{z}_i^{(t)}, p_i) \tag{17}$$

$$l_i = \overline{shift}(m_i, p_i) \prod_{j=1}^{N_i-1} (1 - \overline{shift}(m_j, p_j)), \tag{18}$$

where $\odot$ denotes element-wise multiplication, and $\overline{shift}(x,p)$ denotes spatially shifting the values of x in the direction of p.

Multi-Layer Generation We introduce the Multi-Layer Sampler that matches our diverse layout and semantic simulation. In contrast to SceneDiffusion [25] which scrambles the reference layouts randomly, we sample additional N_l layouts and semantics by the proposed SL-VAE. On the one hand, the SL-VAE ensures that the generated scene layout is reasonable. On the other hand, we take full advantage of the paired object-level *(layouts, semantics)*. Specifically, the denoising scheme consists of four steps:

(a) Sampling additional N_l layouts $\{\mathcal{B}_n = \{b_{n,i}\}_{i=1}^{N_o}\}_{n=1}^{N_l}$ and semantics $\{\mathcal{S}_n = \{s_{n,i}\}_{i=1}^{N_o}\}_{n=1}^{N_l}$ by the proposed SL-VAE. Note that N_l fixed seeds exist for the same scene graph. According to the description of the layered representation, we convert the layout to get offset $\{\mathcal{P}_n = \{p_{n,i}\}_{i=1}^{N_o}\}_{n=1}^{N_l}$.

(b) Aggregating latent codes from various layers in each scene:

$$\boldsymbol{z}_n^{(t)} = \sum_{i=1}^{N_o} l_i \odot \overline{shift}(\boldsymbol{z}_i^{(t)}, p_{n,i}) \tag{19}$$

(c) Estimating the noise $\hat{\boldsymbol{\epsilon}}_n^{(t)}$ from each aggregated latent code $\boldsymbol{z}_n^{(t)}$ and gets denoised aggregated latent code $\hat{\boldsymbol{z}}_n^{(t-1)} \in \{\hat{\boldsymbol{z}}_1^{(t-1)}, \ldots, \hat{\boldsymbol{z}}_{N_l}^{(t-1)}\}$:

$$\hat{\boldsymbol{\epsilon}}_n^{(t)} = \sum_{i=1}^{N_o} m_{n,i} \odot \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_n^{(t)}, E_{\text{CLIP}}(o_i), b_{n,i}, s_{n,i}, a_i, t), \tag{20}$$

where $m_{n,i}$ is the non-overlapping mask converted by the layout $b_{n,i}$.

(d) Updating the latent code of each layer by computing the weighted average of the N_l aggregated latent code

$$\boldsymbol{z}_i^{(t-1)} = \frac{\sum_{n=1}^{N_l} \overline{shift}(l_i \odot \hat{z}_n^{(t-1)}, -p_{n,i})}{\sum_{n=1}^{N_l} \overline{shift}(l_i, -p_{n,i})} \tag{21}$$

where $\overline{shift}(x,-p)$ denotes spatially shifting the values of x in the reverse direction of p.

A.4 More Ablation Studies

Graph Construction. We conduct ablation for graph construction in Table 6. We investigate the impact of different graph components (i.e., CLIP, Box, and Learnable Embeddings) by turning off each independently. We observe that each component improves the performance, all of which are crucial components presented in our DisCo.

Table 6: **Ablation study** for graph construction.

Graph Type	IS ↑	FID ↓
No CLIP Emb.	20.6	23.9
No Box Emb.	21.7	22.5
No Learnable Emb.	21.9	22.2

Computing Consumption. We demonstrate the impact of our proposed CMA on the computational complexity of the U-Net within the Stable Diffusion, as presented in Table 7. We use Floating Point

Table 7: **Ablation study** for computing consumption.

Method	FLOPs (G)	Params (M)	Time (ms)
SD-v1.5 [20]	677.5	859.4	37.9
DisCo	724.1	875.8	108.3

Operations (FLOPs), the number of parameters (Params), and inference time (Time) to measure computing consumption. The FLOPS and Time metrics are conducted by processing the tensor with a resolution of $2 \times 4 \times 64 \times 64$ on an NVIDIA A100 GPU. Our proposed DisCo significantly improves the controllability of the Stable Diffusion with a tolerable increase in computational cost.

A.5 More Visualization Results

Figure 8 showcases more generalizable generation results under consistency for graph manipulation (i.e., node addition and attribute control) in SG2I task. In Figure 9, 10, and 11, we present more visualization comparisons with the methods conditioned by text, layout, or scene graph, which demonstrates the superiority of our DisCo in terms of generation rationality and controllability.

A.6 Limitations

The proposed CMA injects object-level information into the diffusion model via masks from the layout, effectively mitigating semantic ambiguity and limiting attribute leakage. In scenarios involving object overlap, the proposed CMA inhibits direct interaction between the visual token and the object embedding along with its attributes. Nonetheless, the attribute information from the visual token inadvertently leaks into the overlapping region in subsequent layers. Hence, there may be attribute leakage among the objects, as shown in Figure 7.

Figure 7: **Qualitative limitations** on attribute leakage of overlapping.

A.7 Broader Impacts

We demonstrate the superiority of our DisCo over existing generation methods based on text, layout, and scene graphs, suggesting a potential beneficial influence on the realms of art creation and data synthesis. Nevertheless, there remains a concern regarding the possibility of generating malicious images or infringing copyright.

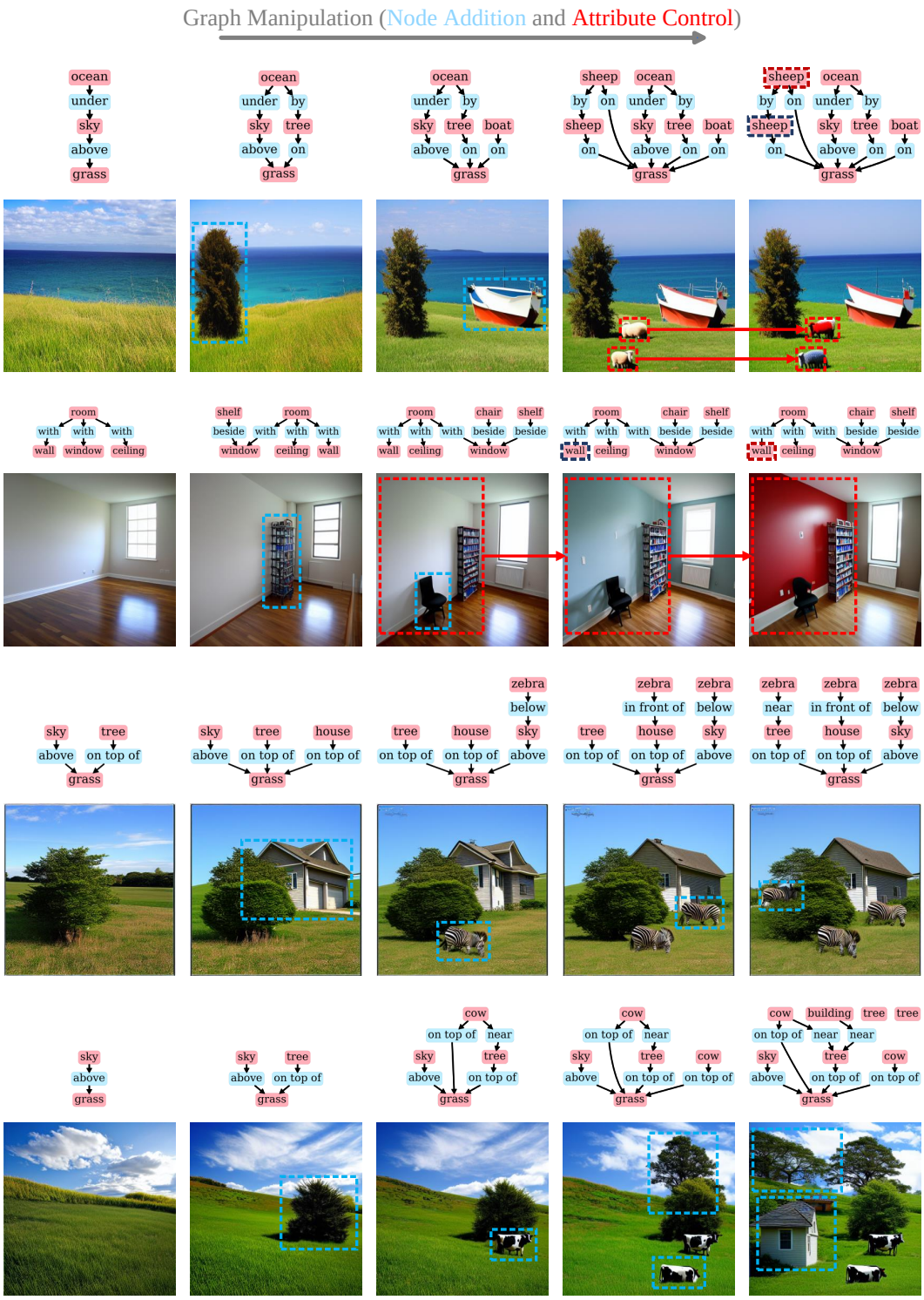

Figure 8: **Generalizable Generation Samples** under Consistency for Graph Manipulation.

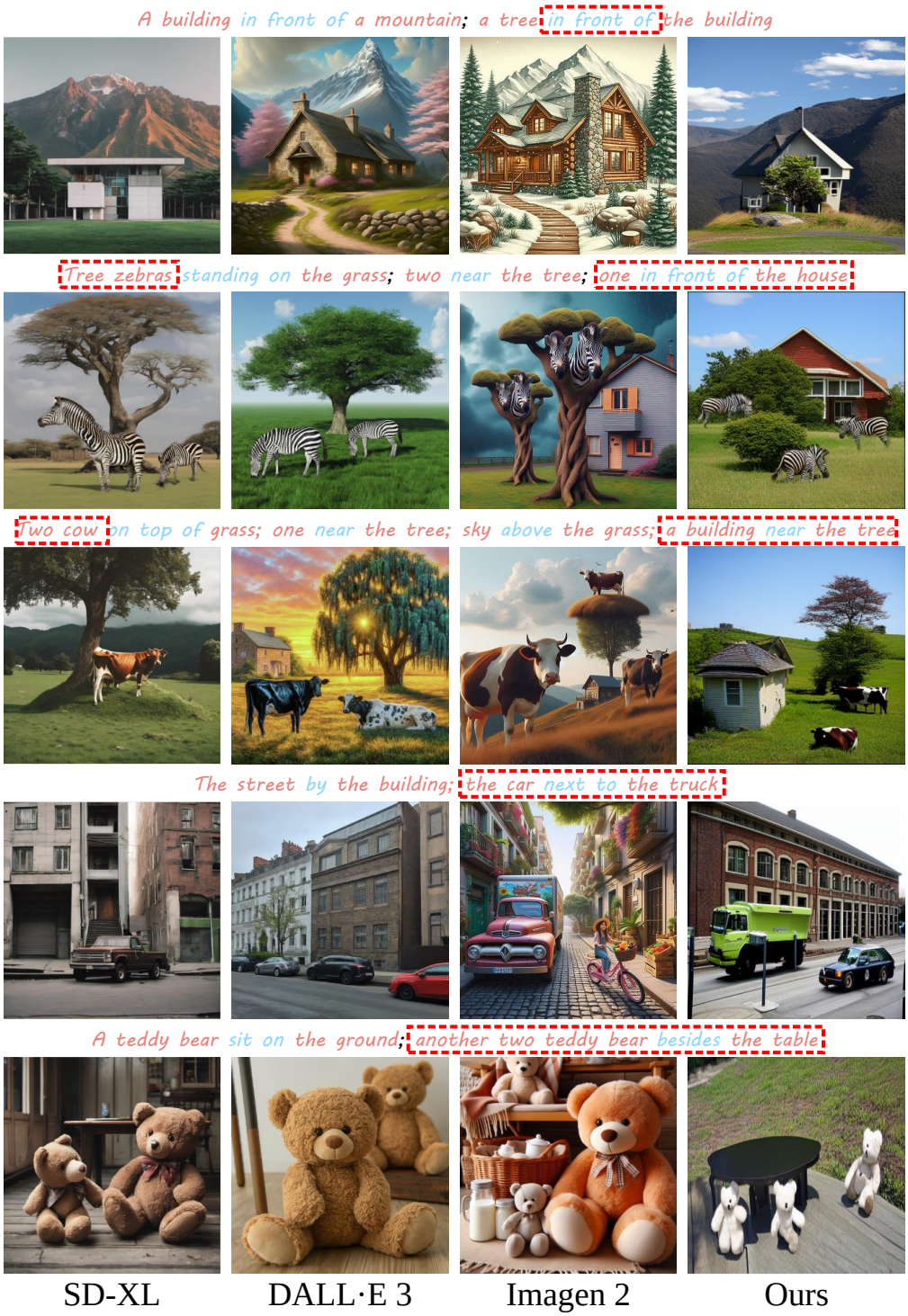

Figure 9: **Qualitative Comparison** with Text-to-Image methods.

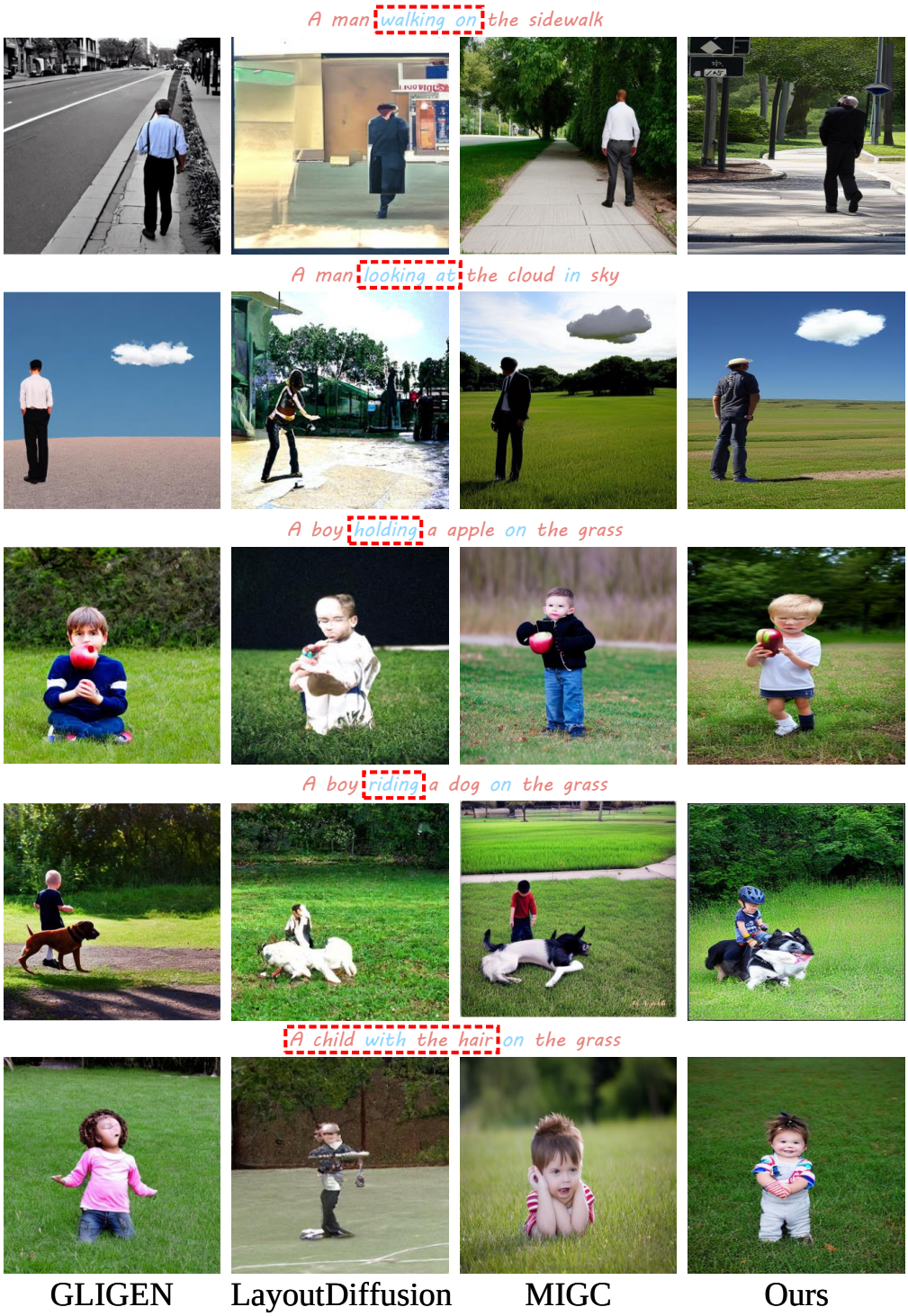

Figure 10: **Qualitative Comparison** with Layout-to-Image methods.

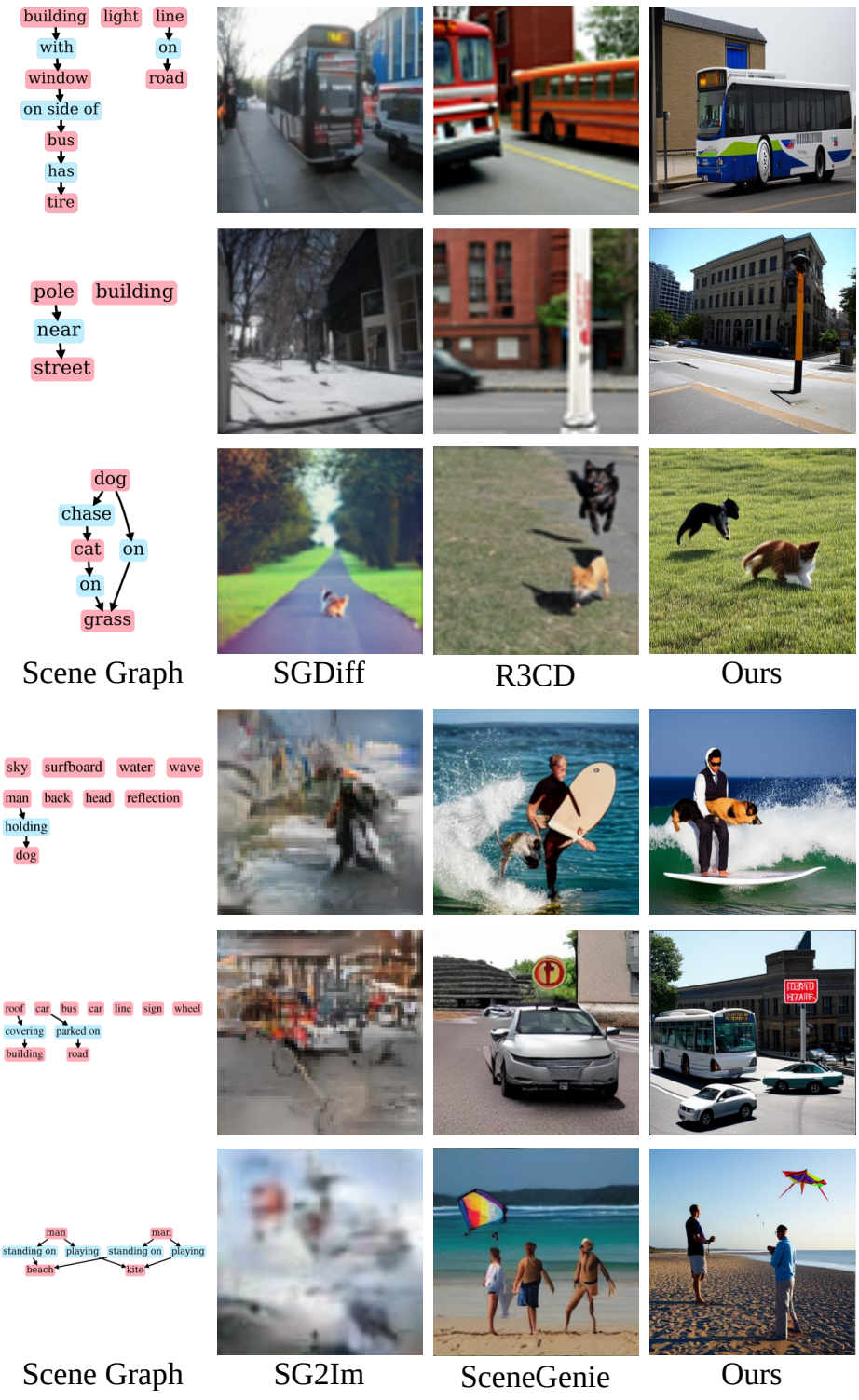

Figure 11: **Qualitative Comparison** with Scene-Graph-to-Image methods.