LaRender: Training-Free Occlusion Control in Image Generation via Latent Rendering - Supplementary Materials

Xiaohang Zhan
Tencent
xiaohangzhan@outlook.com

Dingming Liu
Tencent
dingmingliu3722@gmail.com

1. The RealOCC Dataset

Statistics. As shown in Figure 1, the RealOCC dataset maintains a balanced distribution of images with 2 to 5 objects, capturing diverse occlusion scenarios. Images with more objects usually contain fewer occlusion pairs, which reflects real-world patterns. On average, each image includes 3.56 objects and 3.73 occlusion pairs, offering varied and realistic examples for evaluating occlusion relationships. This distribution ensures the dataset is both diverse and practical for testing occlusion handling methods.

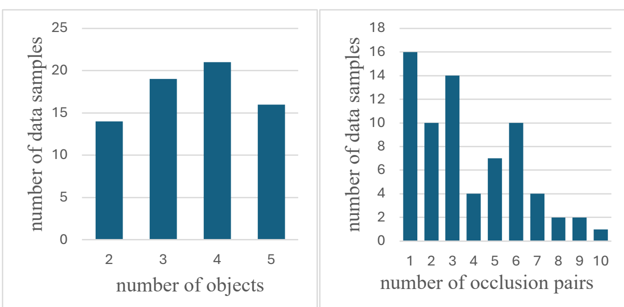

Figure 1. The statistics of the RealOcc dataset.

Examples. Figure 3 shows RealOCC examples with 2 to 4 objects, covering both simple and complex occlusions.

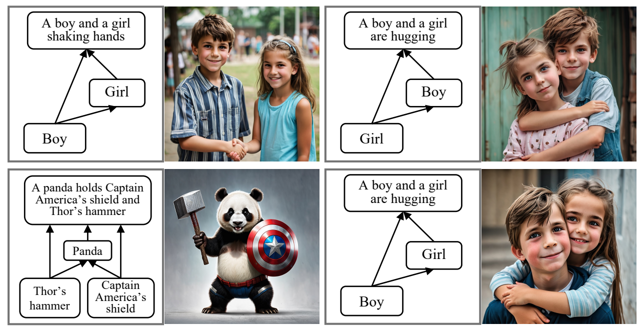

Figure 2. Examples of generating inter-object interactions via background prompts.

2. More Visualizations.

2.1. More results

Figure 6 compares LaRender with SDXL and FLUX. Despite style differences, both handle occlusion well. Failure cases are also shown. Figure 8 highlights a subject occluding various concepts as an interesting example.

2.2. Visualization of Ablation Study

Figure 6 illustrates the effects of different $\sigma_i(t)$ schedules and the use of cross-attention maps. A fixed opaque $\sigma_i(t)$ causes excessive opacity and object disappearance, making it only suitable in early steps. Fixing $\sigma_i(t)$ at a normal value throughout may lead to concept mixing and misaligned occlusions. In contrast, our inverse-proportional schedule generates clearer objects and occlusion relationships. Additionally, cross-attention maps help refine object contours better than bounding box masks alone.

2.3. Visualization of interactions.

Results of inter-object interactions is shown in Figure 2.

2.4. Visualization of 2-element effects.

To showcase LaRender's semantic opacity control, we visualize two elements in Figure 7, showing it can independently control multiple effects and generate high-quality results.

3. LLM instructions

We design the prompt templates in Figure 8 with instructions, in-context examples, and a test case from the user's input. The LLM follows the instruction to identify occlusion pairs and generate object layouts based on occlusion order.

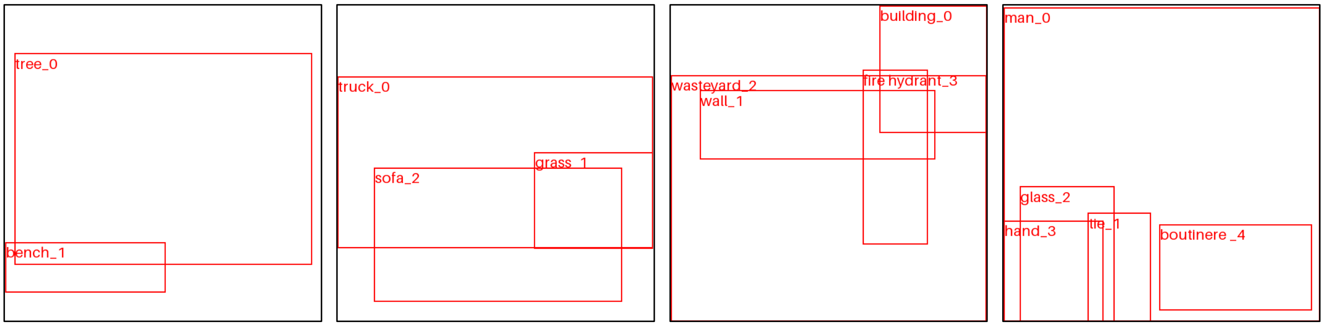

Figure 3. Examples with different number of objects in RealOcc dataset. The bounding boxes are inferred from the amodal masks in the COCOA dataset. The index after the name at the upper-left corner means the ordering from bottom to top. For example, the second example demonstrates grass occludes a truck and a sofa occludes both the track and the grass.

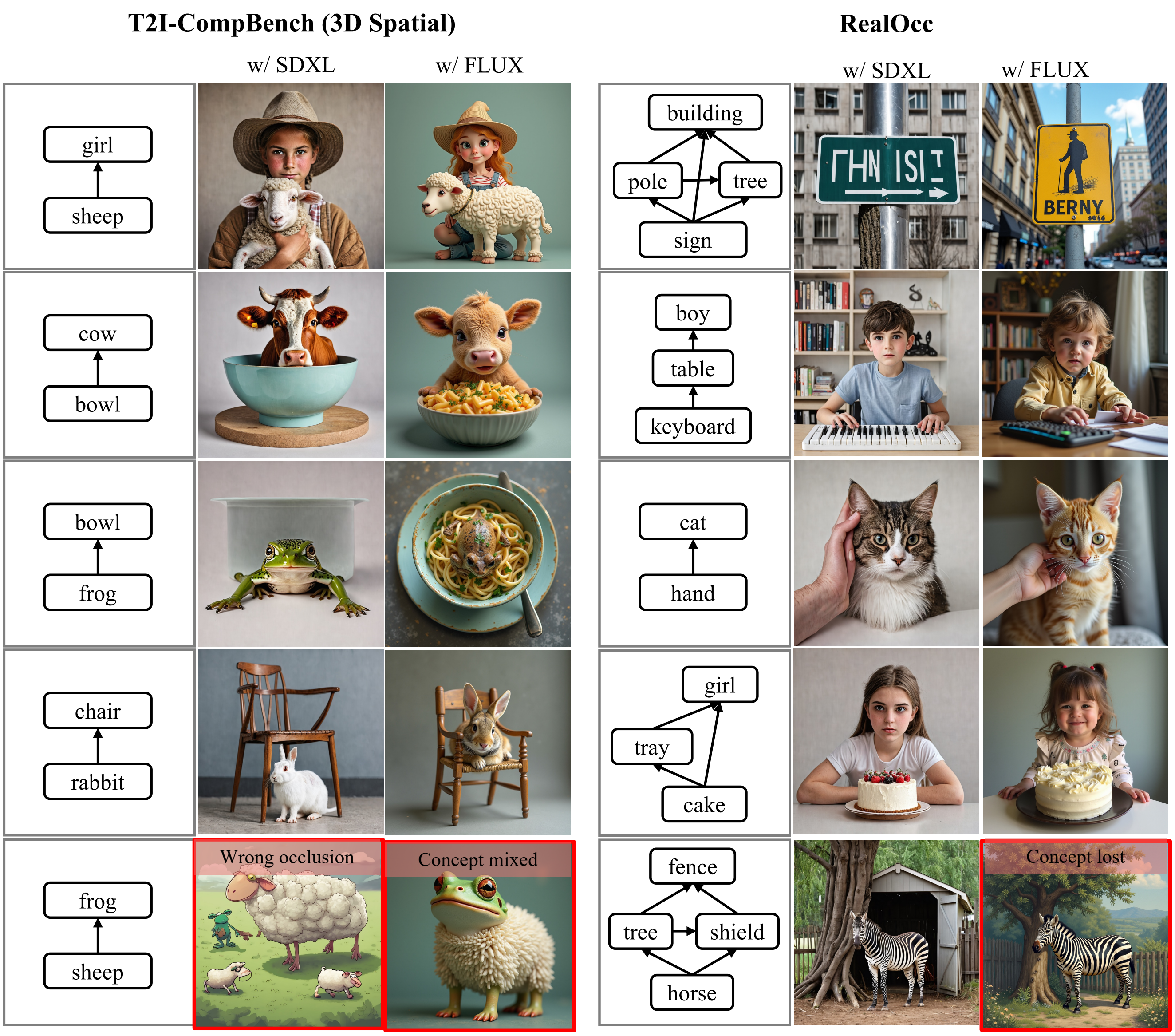

Figure 4. More results by our method, based on SDXL and FLUX. Failure cases are marked in red boxes.

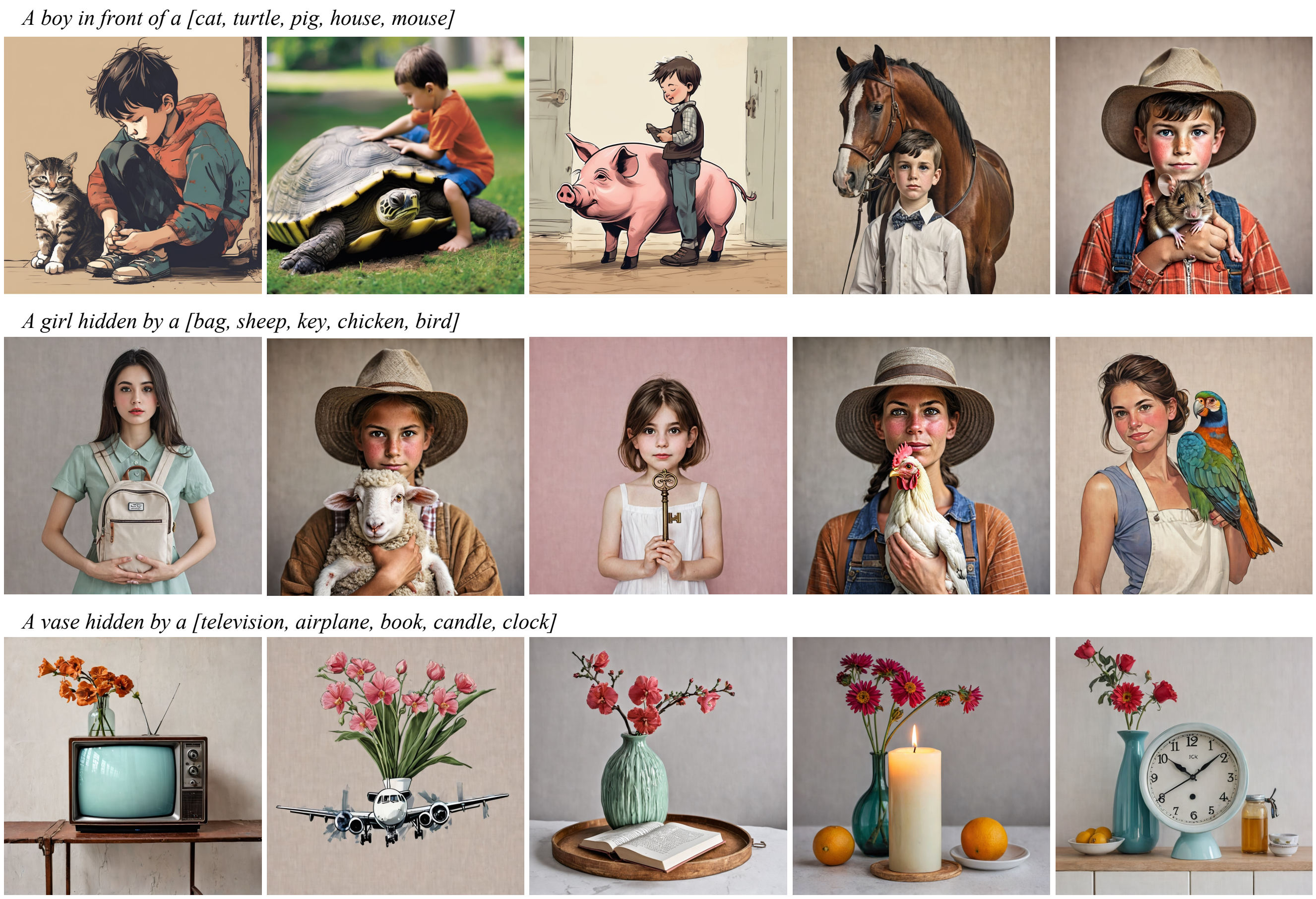

Figure 5. This figure show results of a subject occluding with other objects.

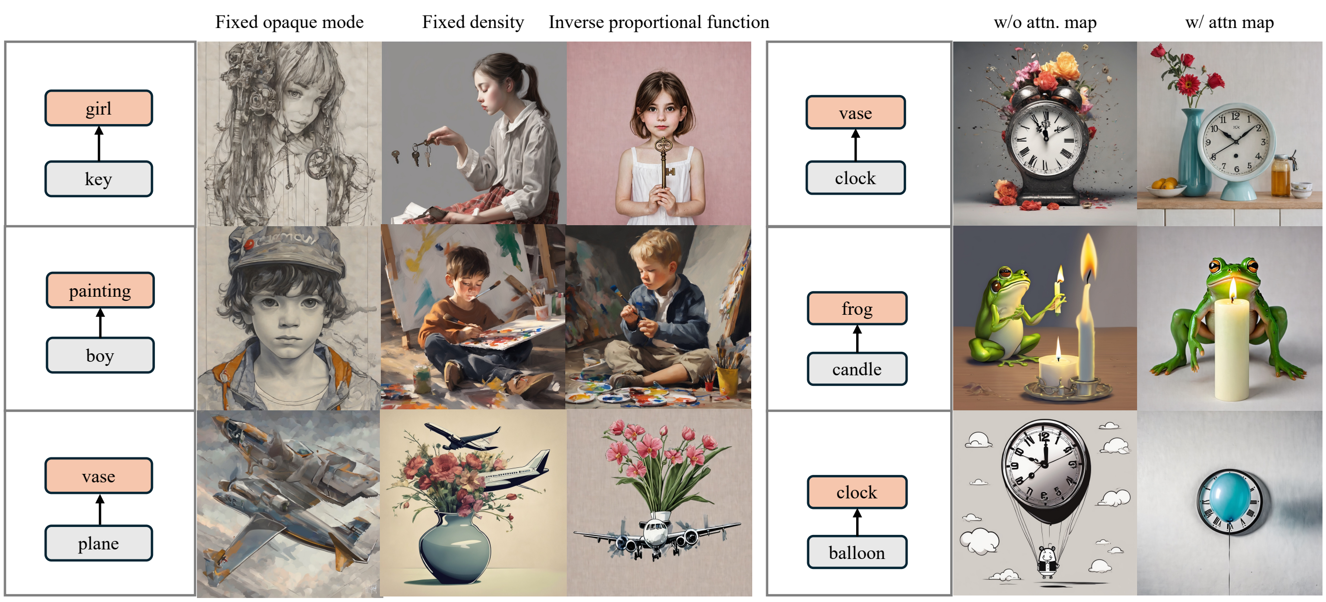

Figure 6. The visual results of ablation study, including the impact of different density schedules (left) and the presence of cross-attention maps in computing transmittance maps (right).

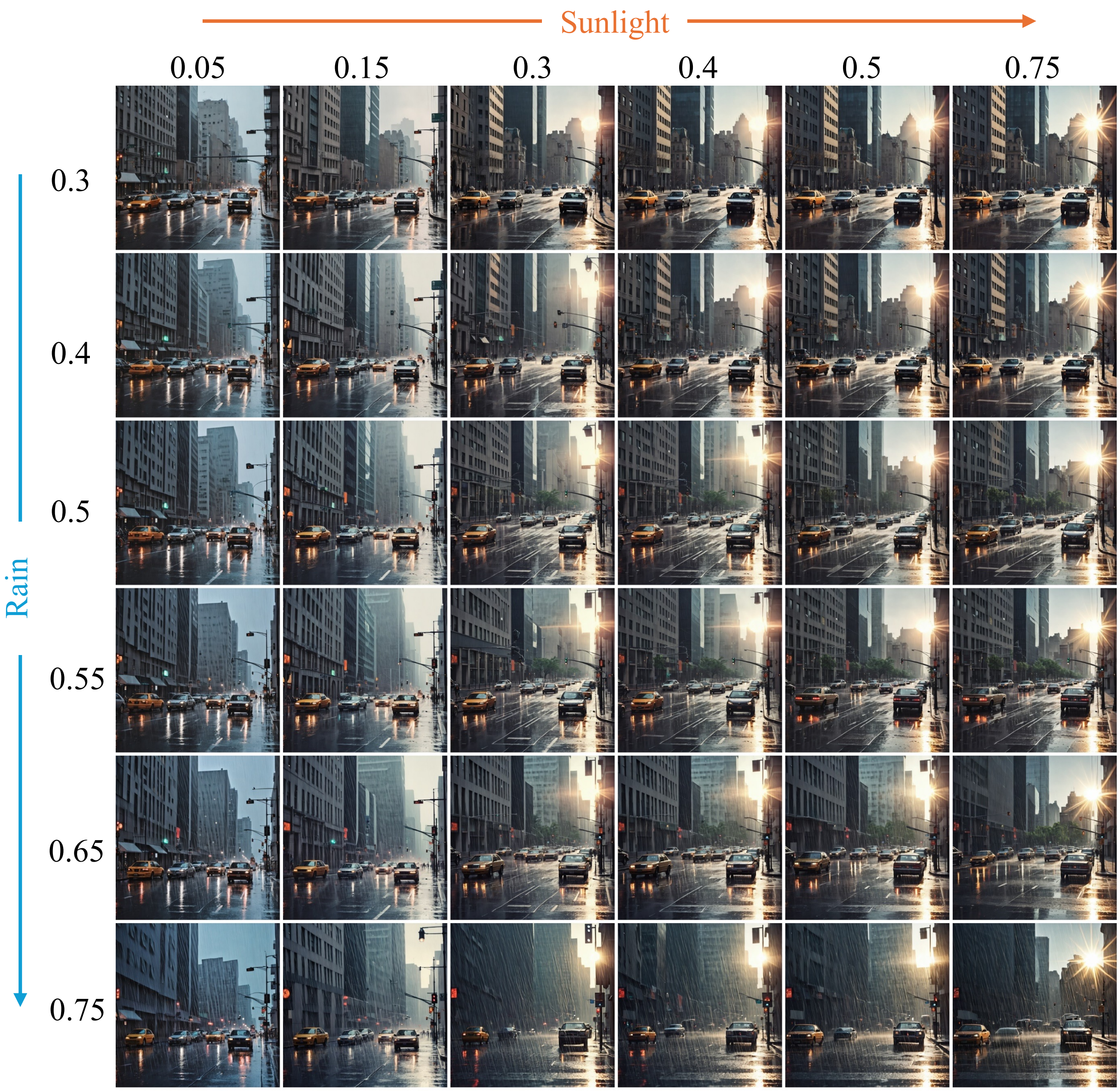

Figure 7. Grid figure of dual element changes. Please zoom in to see the changes.

Instruction

I will provide you with the description of an image. Your task is to parse the description to output the following content:
•prompt: same as input but with double quotation marks
•background: description of the background, if no background, output "NULL"
•objects: a list of descriptions of objects, arranged in the occlusion order from back to front
•locations: a list of the most possible bounding boxes of objects in the format [upper bound, left bound, lower bound, right bound], in the same order as objects
•occlusion_pairs: a list containing all the correct occlusion relationship pairs, e.g., pair (A, B) means A partially obstructs B
Please also strictly follow the rules below:
1.The locations should be reasonable. For pair (A, B), the lower bound of A is usually larger than the lower bound of B, because A is in front of B.
2.The size should be reasonable. For example, a mouse should be smaller than a car.
3.The aspect ratio of objects should be reasonable, and the bounding boxes should adhere to typical shape constraints of real-world objects. For example, a boy or a girl should have a height (upper bound to lower bound) that is greater than their width (left bound to right bound), ideally around 3:1. You need to ensure that the size of each object is not too small, as each object needs to be generated correctly.
4.The locations must satisfy the occlusion relationships described in the image description.
5.The overlap rate of obstructed objects (intersection / area of obstructed object) should be between 0.2 and 0.5. That means, for a pair (A, B), B should be partially obstructed by A and the obstructed area in B should not be too little or too much. For example, for ('a refrigerator', 'a bowl'), even though there is a significant size difference between them, you cannot allow the refrigerator to completely occlude the bowl, as this would result in the bowl being missing.
6.For (A, B), in some special cases, you can choose to have A completely overlap with B, such as in the case of (a clock, a wall). But in this case, it is important to ensure that this occlusion is reasonable and effective.

Examples

Example 1:
Input : a dog behind a sofa
Output:
prompt = "a dog behind a sofa"
background = "NULL"
objects = ["A dog", "a sofa"]
locations = [[0.3, 0.2, 0.7, 0.8],[0.5, 0.1, 0.9, 0.9]]
occlusion_pairs = [("a sofa", "a dog")]

Example 2:
Input : Three teddy bears are standing side by side at the bottom of the living room， from left to right they are blue, yellow, and red. The red teddy bear is positioned in front of the yellow teddy bear, and the yellow teddy bear is positioned behind the blue teddy bear
Output:
prompt = "Three teddy bears are standing side by side at the bottom of the living room， from left to right they are blue, yellow, and red. The red teddy bear is positioned in front of the yellow teddy bear, and the yellow teddy bear is positioned behind the blue teddy bear"
background = "an empty room"
objects= ["a yellow teddy bear", "a red teddy bear", "a blue teddy bear"]
locations = [[0.5, 0.25, 0.85, 0.7],[0.5, 0.55, 0.85, 0.9],[0.5, 0.1, 0.85, 0.45]]
occlusion_pairs = [("a blue teddy bear","a yellow teddy bear"),("a red teddy bear","a yellow teddy bear")]

Test case

Ok, here we go.
Input : On a street, a man and a motorcycle are each obstructing one side of a car

DeepSeek

Output:
prompt = "On a street, a man and a motorcycle are each obstructing one side of a car"
background = "a street"
objects = ["a car", "a motorcycle", "a man"]
locations = [[0.3, 0.3, 0.7, 0.7], [0.4, 0.1, 0.75, 0.4], [0.2, 0.5, 0.9, 0.75]]
occlusion_pairs = [("a motorcycle", "a car"), ("a man ", "a car")]

LaRender

Figure 8. The instruction for DeepSeek to parse the occlusion graph.