A APPENDIX

A.1 MORE IMPLEMENTATION DETAILS.

The Omni3D-ARKitScenes includes 51K training images, 7.6K test images and total 420K oriented 3D bounding boxes annotation. Omni3D-SUN-RGBD includes 5.2K training image, 5K test images and total 40K 3D bounding box annotations. Omni-Indoor is a combination of ARKitScenes, SUN-RGBD and HyperSim datasets. Our geometric ControlNet is trained at a resolution of 256 × 256, following the camera pose annotations provided by the ARKitScenes dataset (Baruch et al., 2021). The novel-view synthesis training is mainly developed based on ControlNet implementation from Diffusers [1]. We use the same training recipe as Diffusers. To report results at a resolution of 512 × 512, we directly input images with a resolution of 512 × 512 into the backbone. Note that different both original ControlNet (Zhang et al., 2023) and DIFT (Tang et al., 2023) use text as input, we input empty text into diffusion model thorough the experiments.

A.2 3D-DETECTION HEAD AND OBJECTIVE.

3D detection head. We use the same 3D detection head as Cube-RCNN (Brazil et al., 2023). Specifically, Cube-RCNN extends Faster R-CNN with a cube head to predict 3D cuboids for detected 2D objects. The cube head predicts category-specific 3D estimations, represented by 13 parameters including (1) projected 3D center (u, v) on the image plane relative to the 2D Region of Interest (RoI); (2) object's center depth in meters, transformed from virtual depth (z); (3) log-normalized physical box dimensions in meters ($\bar{w}$, $\bar{h}$, $\bar{l}$); (4) Continuous 6D allocentric rotation ($p \in \mathcal{R}$); and (5) predicted 3D uncertainty μ. With parameterized by the output of the cube head, then the 3D bounding boxes can be represented by

$$B3D(u, v, z, \bar{w}, \bar{h}, \bar{l}, p) = R(p) \cdot d(\bar{w}, \bar{h}, \bar{l}) \cdot B_{\text{unit}} + X(u, v, z), \tag{9}$$

where R(p) is the rotation matrix and d is the 3D box dimensions, parameterized by $\bar{w}, \bar{h}, \bar{l}$. $X(u, v, z)$ is the bounding box center, represented by

$$X(u, v, z) = \left(\frac{z}{f_x}\right)\left(rx + u \cdot r_w - p_x, \frac{z}{f_y}\right)(ry + v \cdot r_h - p_y), \tag{10}$$

where: $[r_x, r_y, r_w, r_h]$ are the object's 2D bounding box. (f_x, f_y) are the known focal lengths of the camera. (p_x, p_y) represents the principal point of the camera. Given the representation of the 3D bouding box, our detection training objective is

$$L = L_{\text{RPN}} + L_{2D} + \sqrt{2} \cdot \exp(-\mu) \cdot L_{3D} + \mu, \tag{11}$$

where L_{RPN} and L_{2D} are commonly used in 2D object detection such as Faster-RCNN (Ren et al., 2015), here L_{3D} is given by

$$L(u, v)_{3D} = ||B_{3D}(u, v, z_{\text{gt}}, \bar{w}_{\text{gt}}, \bar{h}_{\text{gt}}, \bar{l}_{\text{gt}}, p_{\text{gt}}) - B^{3D}_{\text{gt}}||_1 \tag{12}$$

A.3 TABLE OF LABEL EFFICIENCY

Table 4: Label efficiency.

Methods	Backbone	Pre-training	Tuned Module.	100% data	50% data	10% data
CubeRCNN	DLA34	ImageNet cls.	DLA34+3D Head	31.75	25.32	7.83
DreamTchr	ResNet50	SD distill	Res50+3D Head	33.20	26.61	8.45
DIFT-SD	StableDiff	LION5B gen.	3D Head	28.86	24.94	7.91
3DiffTection	StableDiff+Geo-Ctr	Aktsn nvs.	3D Head	30.16	27.36	14.77
3DiffTection	StableDiff+Geo-Ctr	Aktsn nvs.	Sem-Ctr+3D Head	39.22	35.48	17.11

The label efficiency table is shown in Tab. 4. Please refer to the Experiment section of main text for the analysis of label efficiency experiments.

[1] https://github.com/huggingface/diffusers/blob/main/examples/controlnet/train_controlnet.py

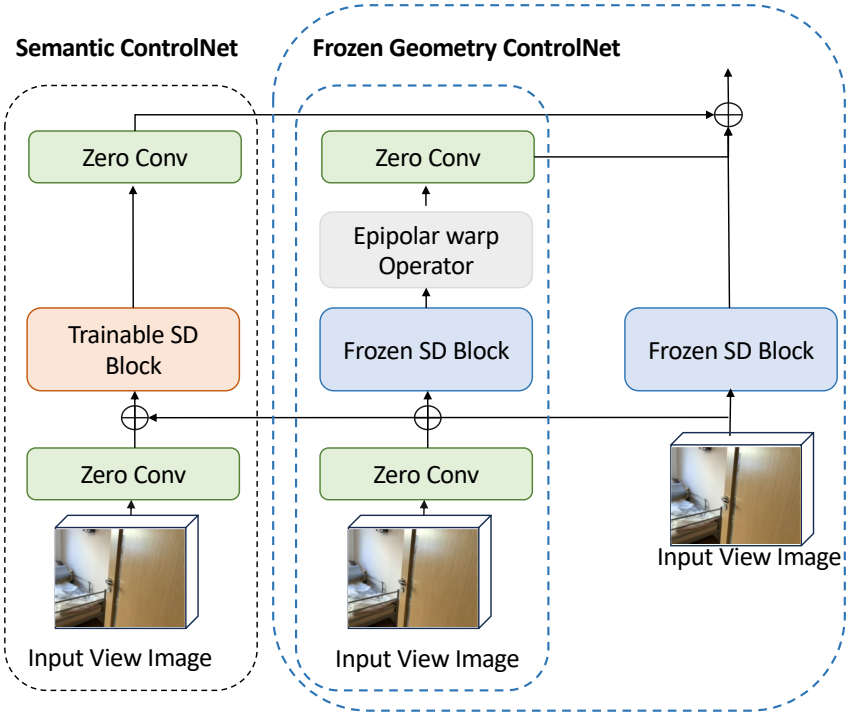

Figure 6: Semantic ControlNet. When tuning Semantic ControlNet, we freeze the pre-trained Gometric ControlNet and the original Stable Diffusion block. For both training and inference, we input the identical pose into the Geometric ControlNet by default.

A.4 Figure of Semantic ControlNet

The figure of Semantic ControlNet is depicted in Fig. 6. Please refer to the main description of Semantic ControlNet in the method part.

A.5 More visualization

Visualization of 3D bounding box. We show more visualization results in this section. Specifically, Fig. A.5 and Fig. A.5 are the 3D bounding box visulization results on the test sets of Omni3D-ARKitScenes and Omni3D-SUN-RGBD, respectively. We observe that our 3DiffTection can predict more accurate 3D bounding boxes compared to Cube-RCNN. More importantly, 3DiffTection does not fail even in very challenging cases, such as the second column of Fig. A.5 and the first column of A.5. They show that 3DiffTection is able to handle occlusions where the chairs are occluded by the tables.

Visualization of novel-view synthesis. We then provide more visualization results about novel-view synthsis, as shown in Fig. A.5. We randomly choose the images that are never seen during the training of geometric ControlNet. To provide how the camera rotate, we present the warped images as well. Even novel-view synthesis at scene level is not our target task, it can be seen that our model can still generalize well under the challenging setting of synthesize novel view from one single image.

A.6 Latency

We evaluate the latency of our proposed method on one RTX 3090 GPU. The latency comparison is shown in Tab. A.6.

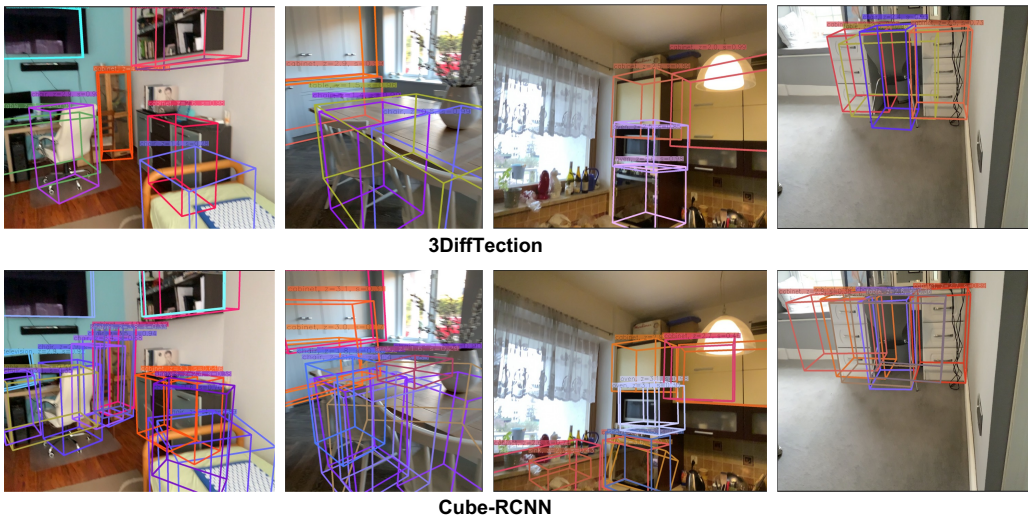

Figure 7: Visualization of 3D bounding boxes on the Omni3D-ARKitScenes test set.

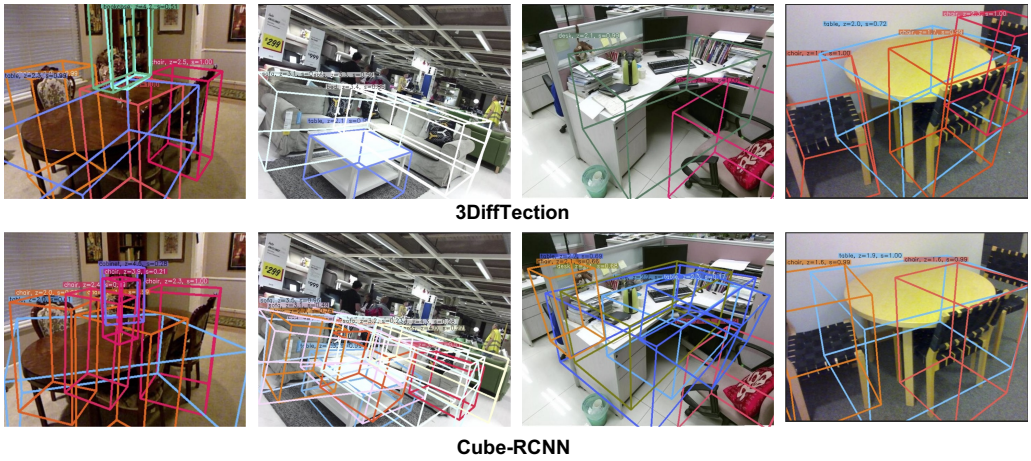

Figure 8: Visualization of 3D bounding boxes on the Omni3D-SUNRGB-D test set.

Method	Latency (s)
3DiffTection (w/o SemanticControlNet)	0.104
3DiffTection	0.133
3DiffTection (w/ 6 virtual view Ensemble)	0.401
Cube-RCNN-DLA34	0.018

Figure 9: Visualization of novel-view synthesis. We rotate the camera by 15 deg anchoring to different axies. The warp image can be used for indicating the camera rotated directions.