Supplementary Material for DiffusionPillars

Anonymous Author(s)
Affiliation
Address
email

In this supplementary document, we present information and experiments in support of the main manuscript. We provide training and architecture details, ablations, and additional comparisons and analyses.

Contents

1 Background: Diffusion Probabilistic Models

In the Method Section of our main paper, we propose using two diffusion models to generate BEV maps and novel view images, respectively. Our formulation follows [19, 8]. From a distribution of scenes we denote a sampled BEV map or a novel view image as x_0, and iteratively add small Gaussian noise to obtain $x_1, x_2 ... x_T$, until x_T approximates an isotropic Gaussian. This forward step is a Markovian fixed process [8, 20] and can be defined as

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad (1)$$

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

where β_t is a variance schedule. In practice, we sample x_t using a closed-form parameterization

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \tag{2}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\epsilon \sim \mathcal{N}(0, I)$.

The goal of each training iteration is to train a model p_θ, often represented by a neural network, that inverts the forward diffusion (i.e., learns the *reverse* diffusion process):

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{3}$$

and

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{4}$$

The reverse process is also Markovian, and we fix the variances Σ_θ. The reverse conditional probability is tractable when conditioned on x_0:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I) \tag{5}$$

We apply Bayes' rule to rearrange the terms and obtain

$$\tilde{\mu}(x_t, x_0) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} x_0 \tag{6}$$

The closed form parameterization of x_t yields

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right) \tag{7}$$

when we represent x_0 as

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_t \right) \tag{8}$$

by rearranging Eq. (2). Thus, we can train our model to predict $\tilde{\mu}_t$, or alternatively, ϵ_t by rearranging the terms. This work predicts $\tilde{\mu}_t$ for generating samples.

During test time, our diffusion model performs generation iteratively as follows

$$z_0 = (f \circ ... \circ f)(z_T, T), \;\; f(x_t, t) = \Omega(x_t) + \sigma_t \epsilon, \tag{9}$$

where $z_T \sim \mathcal{N}(0, I)$, σ_t is the fixed standard deviation at the given timestep, and $\epsilon \sim \mathcal{N}(0, I)$.

2 Training Details

We will open-source all code. All models and baselines were trained on images with a resolution of 256 x 256 pixels. We use a batch size of 16 and a learning rate of 0.0001, as well as the Adam [10] optimizer. All models were trained on a single NVIDIA RTX A6000 with 48 GB memory, and end-to-end training took 48 hours.

2.1 Loss function

The loss function

$$\mathcal{L} = \mathcal{L}_2(\hat{\mathbf{B}}, \mathbf{B}) + \lambda_{reg}\mathcal{L}_{bbox} + \lambda_{dir}\mathcal{L}_{dir} + \lambda_{fg}\mathcal{L}_{cls} + \lambda_{img} MSE\left(\hat{I}, I\right) \tag{10}$$

that is used to optimize the model is a combination of an $\mathcal{L}_2$ loss directly applied on the reconstructed BEV map, a 3D detection loss, and the rendering loss, which is formulated as the MSE between the reconstructed and input image. We set the weights of the different loss components to

$$\lambda_{reg} = 0.0002,\ \lambda_{dir} = 0.00002,\ \lambda_{fg} = 0.0001,\ \lambda_{img} = 0.1$$

While the BEV and the rendering loss are directly applied to the respective outputs, the detection loss uses the reconstructed anchors.

The regression loss is formulated in the reference space. We follow SECOND [23] in the original loss formulation consisting of a box classification loss, box regression loss, and a directional classification loss

$$\Delta x = \frac{x^{gt} - x^k}{d^k}, \Delta y = \frac{y^{gt} - y^k}{d^k}, \Delta z = \frac{z^{gt} - z^k}{h^k}, \Delta w = \frac{w^{gt}}{w^k}, \Delta h = \frac{h^{gt}}{d^k}, \Delta l = \frac{l^{gt}}{l^k},$$
$$\Delta\theta = sin\left(\theta^{gt} - \theta^k\right).$$

The combined box regression is computed as the sum over the SmoothL1 loss of all parameters

$$\mathcal{L}_{bbox,k} = \sum_{\xi_k \in (x,y,z,w,l,h,\theta)} \text{SmoothL1}\left(\Delta\xi_k\right) \tag{11}$$

Each anchor is classified either as one of the $N_{cls} = 3$ classes or, if all values of $\mathbf{v}_{i,j}$ are zero, as the background. Since a large number of anchors are matched to the background we apply the focal loss [13]

$$\mathcal{L}_{cls} = -\alpha_c \left(1 - p^c\right)^\gamma \log p^c \tag{12}$$

with $\alpha = 0.25$ for all classes and $\gamma = 2.0$.

The image-based loss formulation $MSE(\hat{I}, I)$ not only optimizes the image generation, but also the generation of the BEV map and anchors, since it is the only loss that defines the appearance features ξ_k for an anchor k. In the early stages of training, this can have a negative impact on the quality of the BEV anchors, especially in the first iterations. We, therefore, apply linear scaling of 1.3 every 1000 steps to the weight λ_{img} for the first 30k steps before it reaches the value of 0.1.

3 Implementation Details

3.1 Diffusion-based BEV Reconstruction

For the architecture of our diffusion model, we follow DDPM [8] and use a UNet [18] to extract both global and local features of the input observations. The generated output is the one-hot encoded anchor class, 3D box information, directional classifier and appearance feature dimension, resulting in an output dimension of $N_{cls} + 7 + 1 + f_{fg}$, with $N_{cls} = 3$ for CLEVR [9] and $f_{fg} = 32$. The conditioning has the same width and height as the diffused feature map and a channel dimension of 64.

There are 4 downsampling and 4 upsampling layers, with dimensions 64, 128, 256, 512. A connecting layer between the downsampling and upsampling layers has dimensions 512. Each layer contains two ResNet [6] blocks and self-attention [22]. For the timestep, we employ a sinusoidal positional embedding followed by a 2-layer MLP.

To learn correspondences between the inputs (BEV map) and conditions (BEV features), we concatenate the two and employ self-attention. We compute keys (K), values (V), queries (Q) from the concatenated input (X) using learned weight matrices:

$$K = X \cdot W_K;\ V = X \cdot W_V;\ Q = X \cdot W_Q$$

Then, we compute attention weights (A) via softmax of the scaled dot product of Q and K

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d'}}\right)$$

and compute the output Z as the weighted sum of V using weights from A

$$Z = AV$$

where $X \in \mathbb{R}^{n \times d}, W_K, W_V, W_Q \in \mathbb{R}^{d \times d'}, A \in \mathbb{R}^{n \times n}$, and $Z, K, V, Q \in \mathbb{R}^{n \times d'}$. Here, d is the dimensions of the key, value, query vectors.

We fix our forward variances β using a cosine schedule, following [14]. [8] set their variances to be a sequence of linearly increasing constants between $[0.0001, 0.02]$, but the authors of [14] found that the end of the forward noising process was too noisy and could not contribute much to sample quality.

Thus, they employed a cosine schedule so that there is a near-linear drop in the middle of the training timesteps and small changes around $t = 0$ and $t = T$.

$$\beta_t = \text{clip}\left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999\right), \quad \bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2. \tag{13}$$

β_t is clipped at 0.999 to prevent singularities near $t = T$ and the offset s prevents excessively small values of β_t near $t = 0$.

3.2 Diffusion-based Rendering

For simplicity, we use the same diffusion architecture and conditioning mechanisms as in the BEV reconstruction, where the conditioning feature pillar image has the dimension $N_{cls} + f_{fg}$, with $N_{cls} = 3$ for CLEVR [9] and $f_{fg} = 32$, and an output dimension of 3 for the RGB color channels.

3.3 BEV Feature Extractor

We condition the generation of BEV scene layouts using image features that are mapped to the BEV plane. To extract these features, we modify the architecture from CaDDN [17] as follows:

Given an image $\mathcal{I}$, we first extract features $\mathbf{J}(\mathcal{I}) \in \mathbb{R}^{W_F \times H_F \times C}$ using a pre-trained ResNet18 [7], where W_F and H_F are the width and height of the feature grid, and C is the number of feature channels. To reduce memory overhead, we use a 1x1 convolution + BatchNorm + ReLU layer to reduce the number of channels to 64. These features are used to estimate a categorical depth distribution $\mathbf{D}(\mathcal{I}) \in \mathbb{R}^{W_F \times H_F \times D}$, where D is the number of depth bins. For each pixel in $\mathbf{J}$, we predict D probabilities, where each probability represents the confidence that the pixel is in a specified depth bin. The depth distribution is learned by an unsupervised categorical depth distribution network, which modifies the architecture of the semantic segmentation model DeepLabv3 [2]. To modify the semantic segmentation architecture to output pixel-wise probability scores instead of semantic classes, a downsample-upsample architecture is used. We apply an atrous spatial pyramid pooling (ASPP) [1] module with D output channels, which is then upsampled to the original feature size using bilinear interpolation. To normalize the probability distribution, a softmax function is applied to each pixel, creating the categorical depth distribution $\mathbf{D}(\mathcal{I}) \in \mathbb{R}^{W_F \times H_F \times D}$. The D bins are defined using linear-increasing discretization (LID) [21] as follows:

$$d_c = d_{min} + \frac{d_{max} - d_{min}}{D(D+1)} \times d_i(d_i + 1) \tag{14}$$

where d_c is a continuous depth value, d_{min} and d_{max} define the range of depths, and d_i is the discretized depth value.

Using the channel-reduced image features $\mathbf{J}$ and the depth distribution $\mathbf{D}$, we generate a frustum feature grid $\mathbf{F} \in \mathbb{R}^{W_F \times H_F \times D \times C}$ as follows:

$$\mathbf{F} = \mathbf{J}\left(\mathcal{I}\right) \times \mathbf{D}\left(\mathcal{I}\right) \tag{15}$$

The structure of the frustum feature grid $\mathbf{F}$ follows the method proposed in DSGN [3].

Given the camera projection matrix $\mathbf{P}$ in a reference coordinate frame the frustum features $\mathbf{F}$ are then transformed into $\mathbf{V} \in \mathbb{R}^{H_{BEV} \times W_{BEV} \times Z \times C}$, using differential sampling and collapsed into the BEV plane along the z-axis to generate BEV features

$$\mathbf{B} = \mathbf{M} \otimes \mathbf{V}, \text{ with } \mathbf{M} \in \mathbb{R}^{1 \times 1 \times}, \mathbf{B} \in \mathbb{R}^{H_{BEV} \times W_{BEV} \times Z * C}. \tag{16}$$

The differential sampling uses sample points $s_k^f = [u, v, d_i]_k^T$ from $\mathbf{F}$, then convert them to populate $\mathbf{V}$ using $\mathbf{P}$:

$$s_k^f = \mathbf{P} s_k^v, \text{ with } \mathbf{P} \in \mathbb{R}^{3 \times 4} \tag{17}$$

$\mathbf{B}$ is the conditioning BEV feature plane aligned with the grid of anchors, consisting of object or background features.

4 Ablations

Table 1: Monocular 3D Object Detection on CLEVR. **Bold** and underline denote the best and second-best result for each metric for all three ablation.

Method	$AP_{BEV\|R40\|IoU\geq 0.7}\uparrow$	$AP_{3D\|R40\|IoU\geq 0.7}\uparrow$
$\mathcal{L}_{det} + MSE_I$	00.00	00.00
$\mathcal{L}_{2,BEV} + MSE_I$	76.18	46.53
MSE_I	00.00	00.00
Pre-trained Object Detector	90.37	72.47
Full $\mathcal{L}_{2,BEV} + \mathcal{L}_{det} + MSE_I$	**96.41**	**80.57**

We ablate important design and training decisions of our model.

The loss function to train the model in the paper utilizes three different types of losses: 1. an $\mathcal{L}_2$ loss between the underlying ground truth BEV map and the generated BEV map, 2. a regression and classification loss on the extracted object anchors from the BEV layout, following well studied detection pipelines [23, 12, 17] and 3. the MSE loss between the input image and the reconstructed view. We evaluate the necessity of all three loss types in the following by training and evaluating a model for combinations of the loss function.

Loss Ablations. To validate that the loss functions we propose are essential for the method to function, we experiment with combinations of loss components. First, we evaluate the BEV generator only supervised on one of the two losses, either the *BEV loss* $\mathcal{L}_{2,BEV}$ or *Detection Loss* $\mathcal{L}_{det}$ together with the MSE loss on the reconstructed image. A third training of the full model end-to-end is only supervised by the input image, without direct supervision of the BEV. Results in Tab. 1 show that neither the BEV loss nor the detection loss is just enough to train a conditional diffusion model that is able to perform 3D detection. While the $\mathcal{L}_2$ loss is able to generate layouts, the performance is worse. A pure detection loss $\mathcal{L}_{det}$ just generates large object regions, which we visualize in Fig. 1 but fails to reconstruct the scene's layout. Without supervision, self-supervised training is able to use object anchors at arbitrary locations and the background feature to directly leak the input image to the rendering pipeline, not solving for an underlying layout. This is shown in Fig. 2.

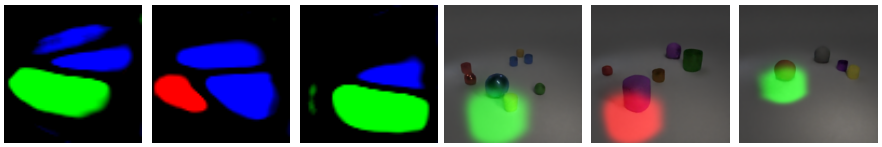

Figure 1: $\mathcal{L}_{det} + MSE_I$**.** We show the reconstructed BEV layout and the respective feature pillars projected onto the generated image of that scene. For the standard detection loss and rendering loss, the training of the diffusion model fails to reconstruct the true underlying layout.

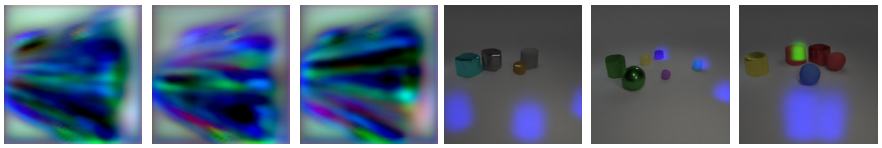

Figure 2: $Self - supervised$**.** We show the reconstructed BEV layout, as well as the feature pillars projected onto the generated image of that scene. Without any supervision of object locations, the generated BEVs reconstruct some latent feature space but do not have any understanding of the underlying scene layout.

BEV Diffusion Pre-training. Further, we also experiment with training the diffusion model without the reconstruction and show the quality of this model as a standalone 3D object detection method in Tab. 1. We find that training with the entire pipeline has a positive impact on the detection results over just training the first half of the model.

5 3D Detection

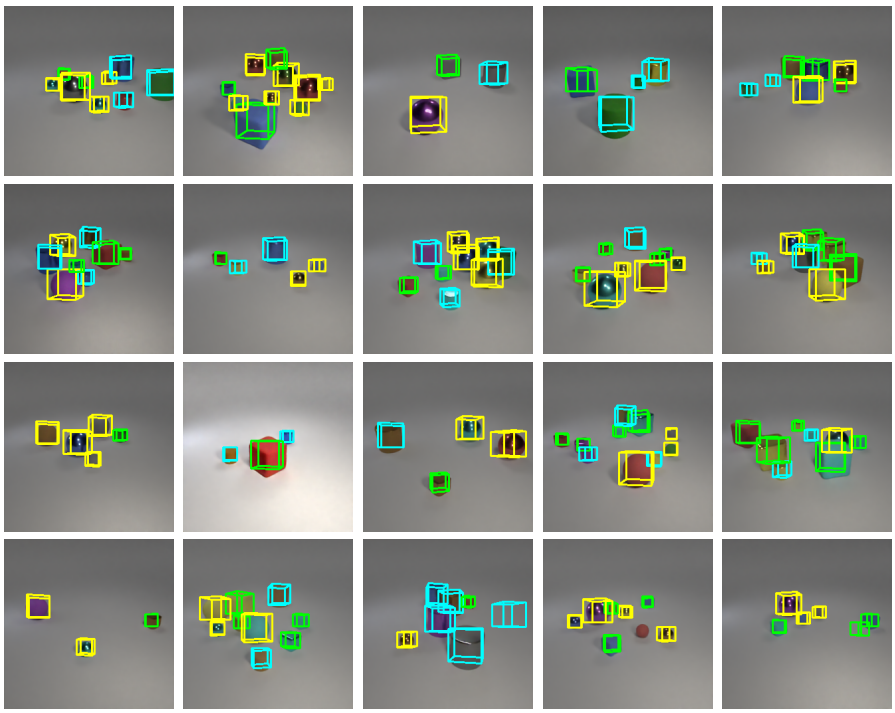

Figure 3: **Visualization of 3D Bounding Boxes for our method.** We visualize 3D bounding boxes, that were generated with OUR proposed methods. For visualization, we apply the tooling from the KITTI dataset [5].

Table 2: Monocular 3D Object Detection on CLEVR. **Bold** and underline denote the best and second-best result for each metric.

Method	$AP_{BEV\|R40\|IoU\geq 0.7}\uparrow$	$AP_{3D\|R40\|IoU\geq 0.7}\uparrow$
DD3D	**98.20**	<u>83.23</u>
DEVIANT	94.86	**86.08**
MonoFlex	95.97	82.20
CaDDN	92.46	79.41
DiffusionPillars (ours)	<u>96.41</u>	80.57

We present object detection results from our method as well as a comparison with four state-of-the-art monocular 3D object detection methods. The evaluated methods are as follows:

DD3D [16] is an end-to-end, scalable detector that does not suffer from the limitations of previous pseudo-lidar methods, such as overfitting and increased complexity.

DEVIANT [11] uses convolutional blocks that are equivariant to 3D translations, including depth translations in the projective manifold. This allows for consistent depth estimation, leading to better 3D object detection results.

MonoFlex [24] optimizes truncated and non-truncated objects separately, and estimates depths from different groups of keypoints.

CaDDN [17], mentioned previously in Section 3.3, combines extracted depth-aware BEV features with the detection head of PointPillars [12], a LiDAR-based object detection method.

As seen in Tab. 2 and visualized in Fig. 3, our method demonstrates object detection results on par with traditional feed-forward 3D object detection architectures, with the second-best results in BEV average precision. While BEV precision is relatively high, 3D precision is still on par, but suffers

slightly, specifically in the cube class. We hypothesize that this is due to the difficulty in accurately predicting and generating rotation.

6 Additional Layout Reconstruction and Manipulation

We present additional manipulation and reconstruction results from our approach. All scenes displayed in the following figures are either reconstructed from randomly sampled scenes of the 5.5k test scenes or objects from those reconstructions randomly sampled in new compositions. In some cases, objects are not detected, but we still include those scenes and manipulation results.

Scene Manipulation. Given an initial layout of a scene we add random offsets in the BEV to one or all objects. In addition to the results shown in the main paper, we present renderings of novel scene layouts in Fig 4. This can also be used to control renderings through user-defined inputs. Occlusions are handled by our method as well as in the reconstruction task.

Novel View Synthesis. In addition to the images shown in the main paper, we present results on the novel view synthesis task in Fig 5. Cameras are rotated along the scene at different heights. In addition to novel views, we show the input view and a reconstruction of the same view.

Detection & Reconstruction. In Fig. 6 we present additional reconstructed images from the test set, as well as outputs from the diffusion-based detection as BEV maps, and visualizations of the pillars that condition the image generation process. In the BEV and pillar images each color (RGB) represents one of the three classes in the CLEVR dataset: Red = Cube, Green = Sphere, Blue = Cylinder. The overall reconstruction quality is satisfactory, but sometimes the diffusion probabilistic process leads to small errors in the reconstructed images, including wrong material properties or reconstructed color, rounded corners and edges at cubes, and inconsistent lighting.

Reconstruction from Inverse BlobGAN. Prior generative methods with intermediate layout representations usually do not offer a straightforward reconstruction approach that is not trained afterward and maps from the image space back to the latent space. BlobGAN [4] proposes such a method, with published code. Unfortunately, this method only supports images with a fixed number of objects in all scenes. We made some modifications to the code to allow for reconstructions from generated images with arbitrary, but known numbers of objects in the generated image. Results are presented in Fig 7.

Novel Scene Composition. By removing or adding objects to the intermediate layout, we are able to create novel scene compositions, which are shown in Fig 8. Using the pillar projections, our diffusion-based rendering approach is able to generate images of more complex novel scenes, including occluding objects.

Novel Scenes from GIRAFFE. We compare results to the results from similar experiments on GIRAFFE [15] in Fig. 9 applying the provided rendering script. The overall rendering quality suffers from blurred features and changes in object appearances after adding new, occluding objects to the scene.

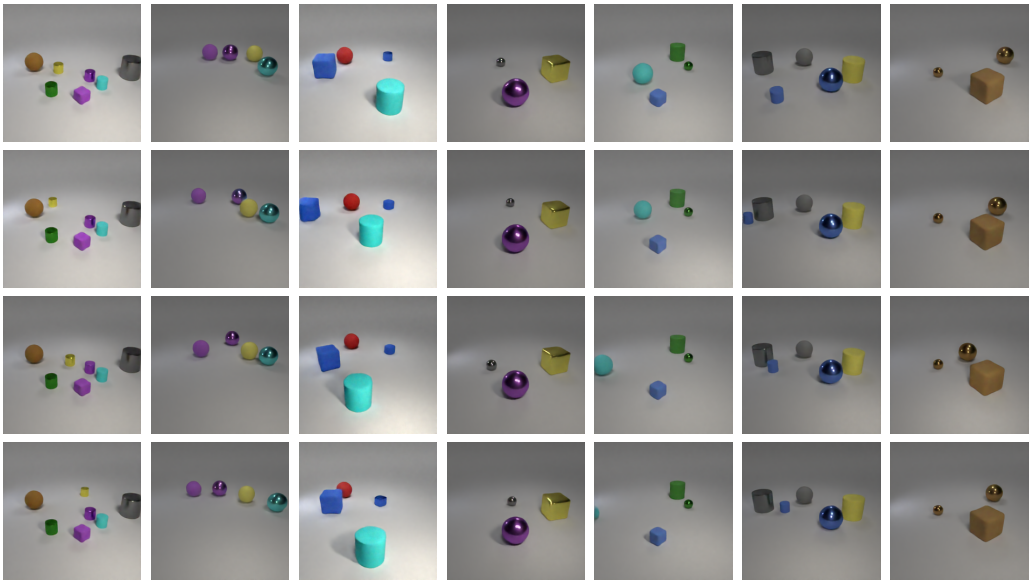

Figure 4: **Object Translation.** We present renderings from layouts where we translate objects on the BEV. We first sample a random scene and add offsets to all or a single object in the scene. Our scene layout is able to handle complex novel layouts and occlusion.

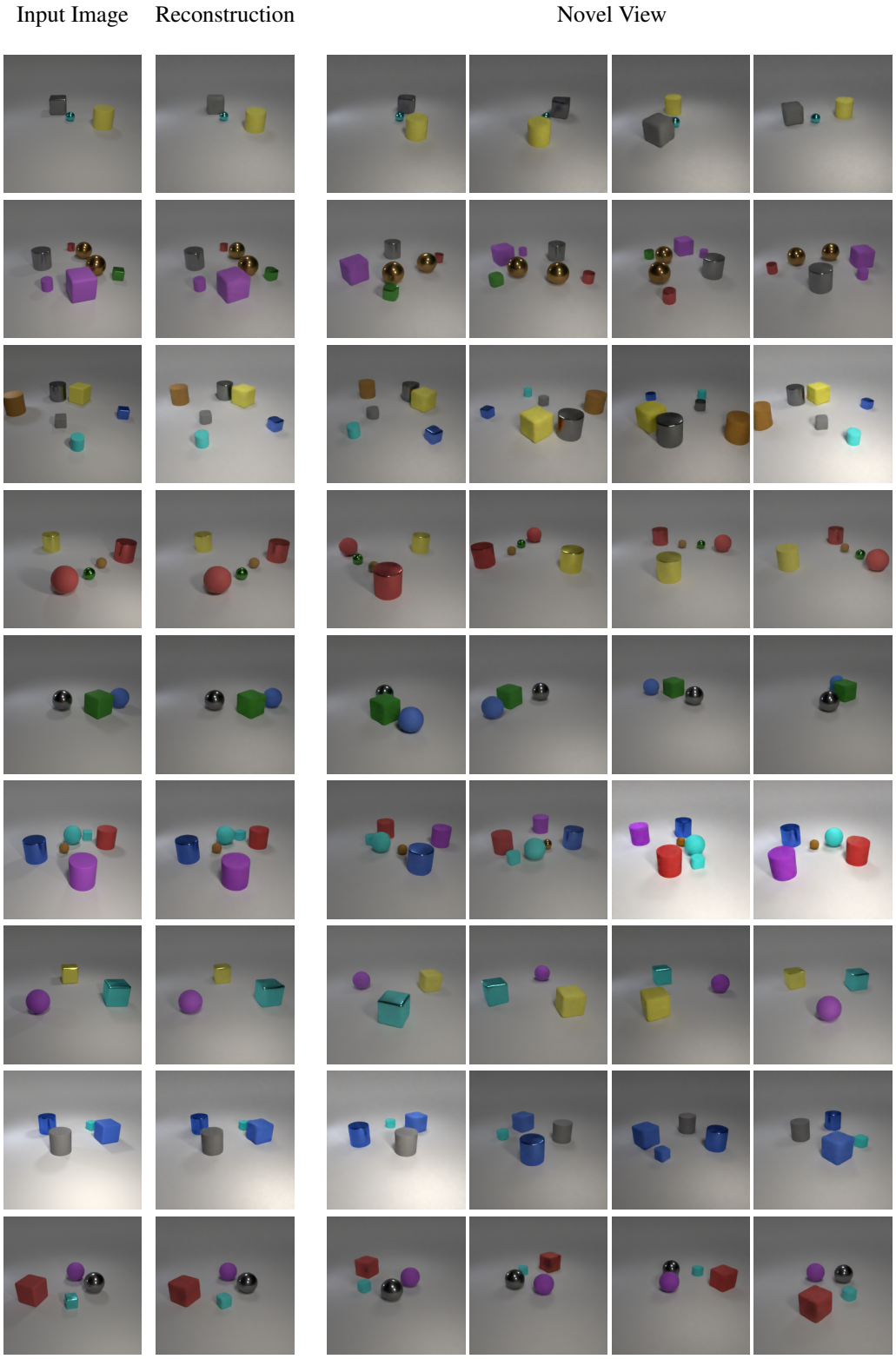

Figure 5: **Novel View Synthesis.** We first reconstruct a scene from an input image. Next to this is a rendering of the reconstructed scene. The four images on the right show novel views, which are generated through manipulation of the viewing angle of the camera.

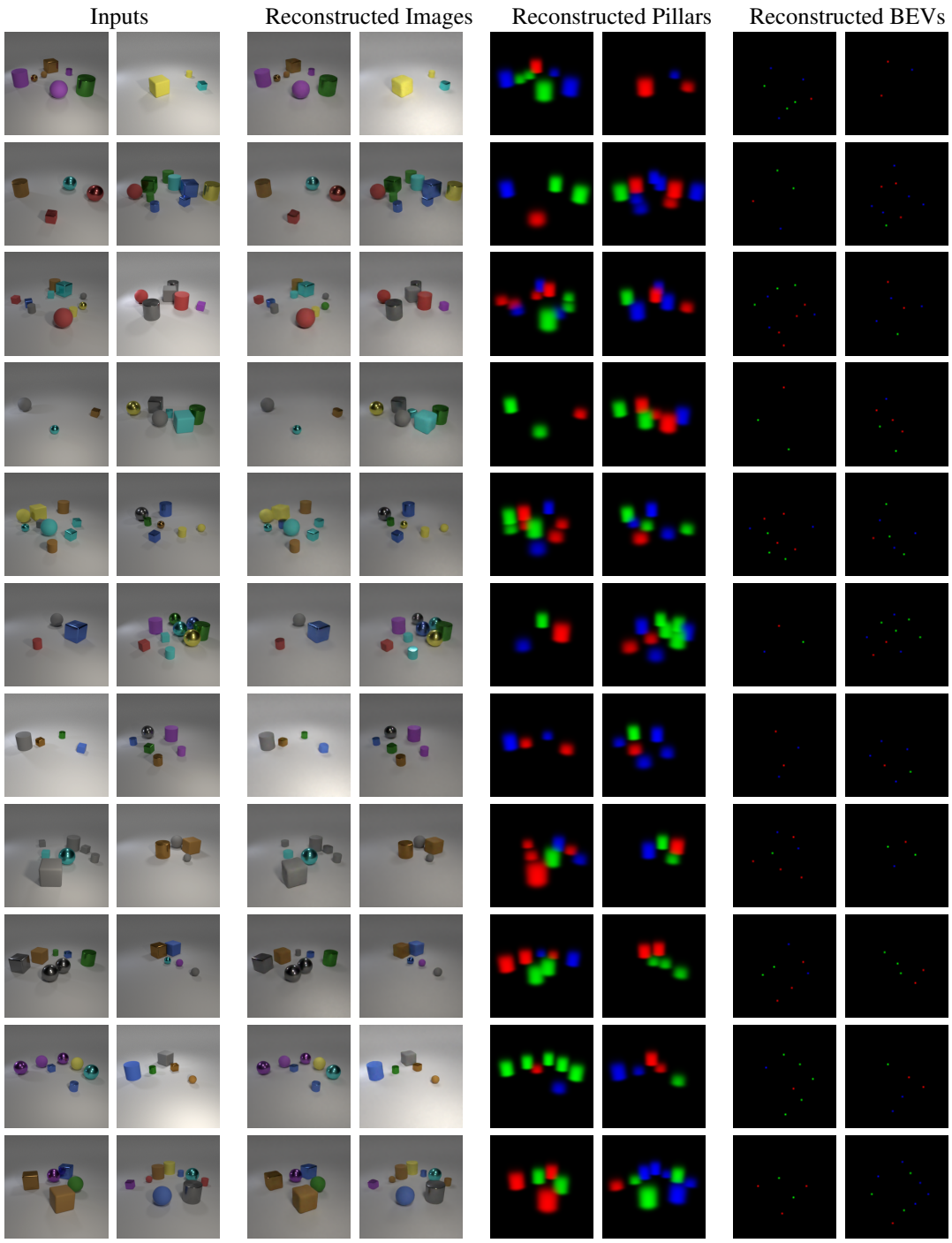

Figure 6: **Reconstructed Scenes.** Images of the scene are reconstructed conditioning our method on the left input images. The BEV map is the output of the diffusion-probabilistic BEV generation method. The images of the pillars are directly projected from the 3D reconstruction of the scene proposed by us. Red = Cube, Green = Sphere, Blue = Cylinder.

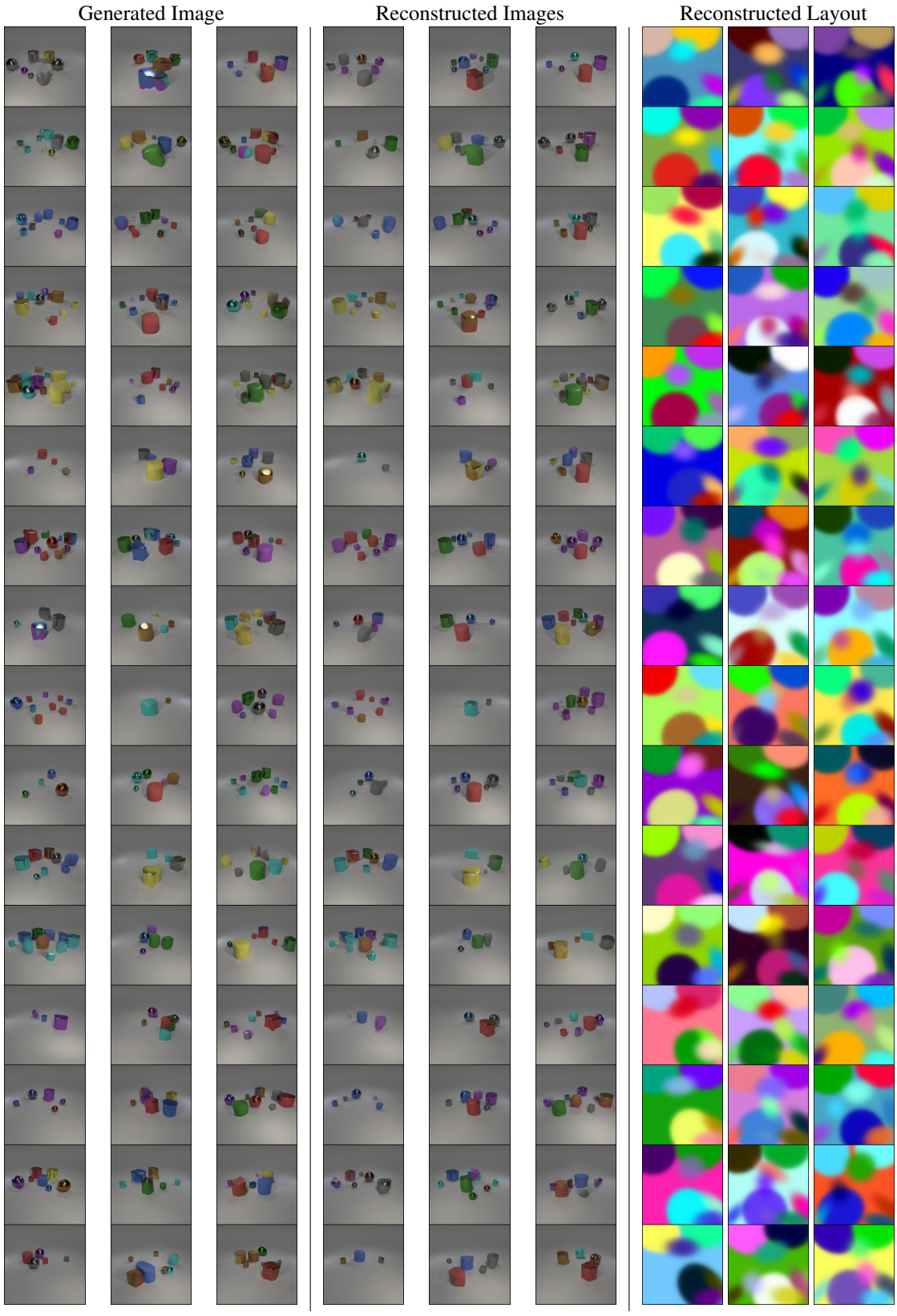

Figure 7: **Inverse BlobGAN.** [4] We visualize results from BlobGAN's inverse path, as well as the intermediate blob layouts. For those experiments we chose to use BlobGANs own reconstruction over GT images to handle a different amount of objects in each experiment.

Figure 8: **DiffusionPillars (Ours) Scene Composition.** (top-down) We add or remove objects to a scene. Lighting changes are the result of diffusion-based rendering in unseen scenarios. The model was only trained on a dataset with 3 to 10 objects but is able to reconstruct scenes with fewer and more objects.

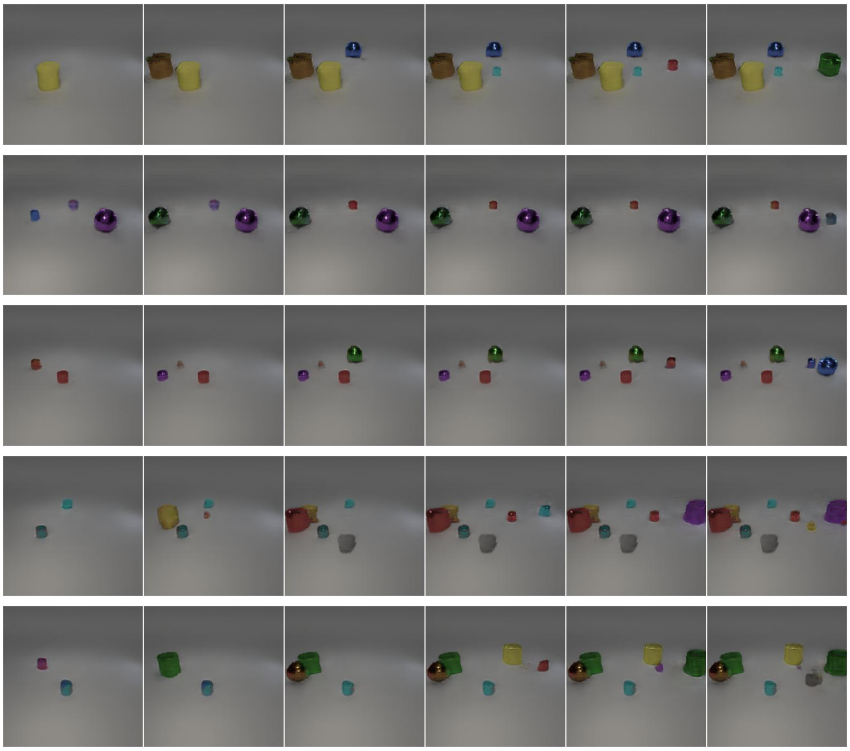

Figure 9: **GIRAFFE [15] Scene Composition.** (left-right) We visualize results from GIRAFFE's scene manipulations through sampling objects in a scene. For those experiments, we chose to use the code for rendering novel views and scene manipulation provided by the authors of GIRAFFE.

References

sorting

[1] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv e-prints*, arXiv:1706.05587 (June 2017), arXiv:1706.05587. DOI: 10.48550/arXiv.1706.05587. arXiv: 1706.05587 [cs.CV].

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *CoRR* abs/1706.05587 (2017). arXiv: 1706.05587. URL: http://arxiv.org/abs/1706.05587.

[3] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. "Dsgn: Deep stereo geometry network for 3d object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12536–12545.

[4] Dave Epstein, Taesung Park, Richard Zhang, Eli Shechtman, and Alexei A. Efros. *BlobGAN: Spatially Disentangled Scene Representations*. 2022. arXiv: 2205.02837 [cs.CV].

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[9] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2901–2910.

[10] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015).

[11] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. "DEVIANT: Depth EquiVarIAnt NeTwork for Monocular 3D Object Detection". In: *In Proceeding of European Conference on Computer Vision*. Tel-Aviv, Israel, Oct. 2022.

[12] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. "Pointpillars: Fast encoders for object detection from point clouds". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12697–12705.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[14] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8162–8171.

[15] Michael Niemeyer and Andreas Geiger. "GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[16] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. "Is Pseudo-Lidar needed for Monocular 3D Object detection?" In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[17] Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. "Categorical depth distribution network for monocular 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8555–8564.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[19] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[20] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[21] Yunlei Tang, Sebastian Dorn, and Chiragkumar Savani. “Center3D: Center-based Monocular 3D Object Detection with Joint Depth Understanding”. In: (May 2020).

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[23] Yan Yan, Yuxing Mao, and Bo Li. “Second: Sparsely embedded convolutional detection”. In: *Sensors* 18.10 (2018), p. 3337.

[24] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. “Objects Are Different: Flexible Monocular 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 3289–3298.