MonoPlace3D: Learning 3D-Aware Object Placement for 3D Monocular Detection

Supplementary Material

Contents

A. Additional placement results

A.1. Quantitative evaluation

To quantify the performance of placement, we compute the following three metrics on the training set of KITTI: **1) Overlap:** As road regions can cover most of the plausible locations for cars, we evaluate the predicted location by checking whether the *center of the base of the 3D bounding box* is on the road. Specifically, we compute the fraction of boxes that overlap with the road segmentation obtained using [8]. **2)** $\theta_{\mathbf{KL}}$: We evaluate the KL-divergence between the distribution of orientation of the predicted 3D bounding box and the ground truth boxes. We present quantitative results in Tab. 1, where our method achieves superior overlap scores, suggesting the superiority of placement.

Table 1. Ablation over SA-PlaceNet components

Method	Random	w/o var & geo	w/o geo	w/o var	Ours
Overlap ↑	0.20	0.15	0.17	0.35	**0.36**
θ_{KL} ↓	1.37	0.66	1.18	0.32	**0.30**

A.2. Placement on nuScenes [1] dataset

We validate the generalization of our method by training SA-PlaceNet on a subset of a recent driving dataset - NuScenes [1] in Fig. 1. We visualize predicted 3D bounding boxes and realistic renderings from our method. Our approach produces plausible placements and authentic augmentations for the given scene.

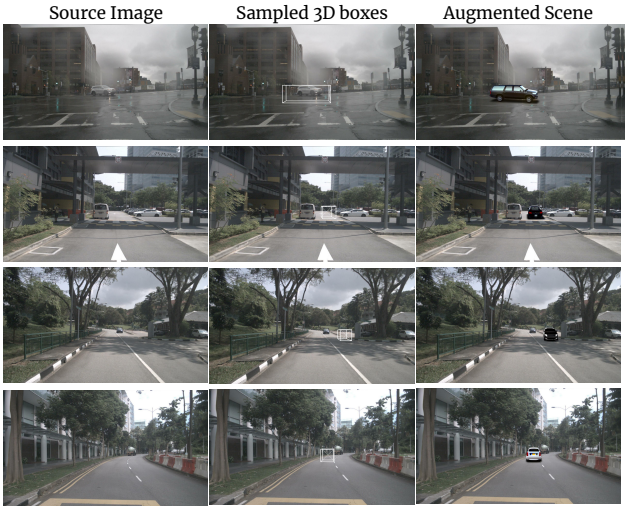

Figure 1. Placement on nuScenes [1] dataset.

A.3. Controlling traffic density in scenes

Our augmentation method enables us to control the traffic density of vehicles in the input scenes by controlling the number of bounding boxes to be sampled. We present results for generating low-density ($1-3$ cars added) and high-density ($3-5$ cars added) traffic scenes in Fig. 2.

A.4. Placing other categories

Our method enables us to learn placement for other categories from KITTI datasets. Specifically, we trained a joint placement model to learn the distribution of 3D bounding boxes for cars, pedestrians, and cyclists. To render the pedestrians and cyclists, we leverage simple copy-paste rendering as discussed in Sec. F.1. We present placement results in additional categories in Fig. 3. The pro-

Figure 2. Augmented training dataset for 3D object detection: Given a sparse scene with few cars, we place cars at the predicted 3D bounding box locations using our rendering algorithm. We present two sets of results, one with low density ($1-3$ cars added) and another with high density ($4-5$ cars added) for each scene.

posed method predicts plausible locations, orientation, and shape of the object, enabling rich scene augmentations. Using these augmentations for training leads to significant improvement in performance for less frequent cyclist and pedestrian categories (Tab.3 main paper).

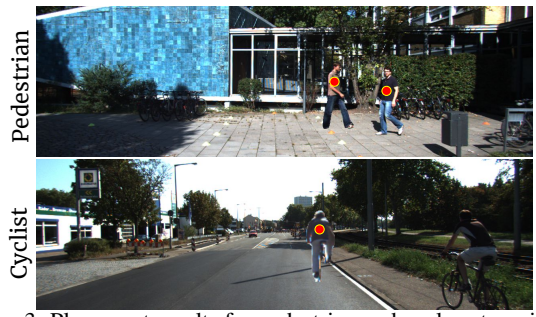

Figure 3. Placement results for pedestrian and cycle categories on KITTI dataset. Note that we applied copy-paste in the predicted 3D object box locations to generate the augmentations. Though copy-pasting causes image artifacts, these augmentations still improve 3D detection performance, as shown in the main paper.

B. Additional object detection results

B.1. Monocular 3D detection in indoor scenes

Our proposed method is generalizable for 3D detection in indoor environments. To demonstrate this, we performed a preliminary experiment involving monocular 3D detection on SunRGBD [21] dataset. We adapt our placement network building on an indoor detection network - ImVoxelNet [15]. We used copy-paste along with the predicted object locations to generate data augmentations.

Table 2. Indoor 3D detection

config.	mAP@0.25
ImVoxelNet - w/o 3D augm.	0.410
ImVoxelNet - w ours	**0.430**

The generated augmentations are highly effective and improve upon the monocular 3D detection performance, as shown in Tab. 2. This indicates the superior generalization of our method for diverse environments. We believe a detailed exploration of our work for indoor environments is a promising future direction.

B.2. Improving 2D object detection

As our approach provides consistent 3D augmentations, it also enables to improve the performance of 2D object detectors. Specifically, our placement model also predicts the 2D bounding box along with the 3D bounding box (followed in most of the 3D detection works). We use these predicted 2D bounding box annotations to obtain a labeled 2D detection dataset. We evaluate the gains from our augmentations on 2D object detection on off-the-shelf 2D detector CenterNet [22] in Tab. 3. Following [16], we use a standardized approach to report AP_{40} metric instead of the AP_{11} for evaluation. Notably, our proposed augmentation method, though designed for 3D detection, can also improve the performance of 2D object detection, proving the task generalization of the proposed approach.

Table 3. 2D Detection Performance on '*Car*' category with CenterNet [22]

config.	AP2D@IOU=0.5		
	Easy	Mod.	Hard
w/o 3D Aug.	86.03	73.74	65.08
Ours	**89.56**	**76.79**	**72.28**

B.3. 3D object detection on BEV based detector

Our method generalizes to BEV-based detection, as our placement model predicts 3D bounding boxes in the world coordinate space. We train BEV-based **DeTR3D** on multi-view nuScenes, augmenting individual camera views by placing our ***2D car renderings*** in non-overlapping image regions. Since overlapping regions are mostly confined to the peripheries of adjacent camera views [11], our augmentations effectively improve detection performance (Tab. 4). For overlapping image regions, a possible solution is to use 3D cars and render consistent multi-views for placement.

Table 4. Detection on BEV based 3D detector DeTR3D

config.	NDS	mAP
Detr3D - w/o 3D augm.	0.434	0.349
Detr3D - w ours	**0.451**	**0.381**

B.4. 3D object detection on MonoDETR [19]

To validate the generalizability of our approach, we evaluate proposed 3D augmentation on a recent 3D monocular detection model MonoDETR [19] on the KITTI dataset in Tab. 5. We report the baseline results without our augmentation from the original paper. Our method consistently outperforms the baseline in all three settings. The compre-

hensive evaluation across several detectors (also in the main paper) evidently shows the generalization of our proposed 3D augmentation method.

Table 5. 3D Detection Performance on Car with MonoDETR [19]

MonoDETR	3D@IOU=0.7			3D@IOU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard
w/o 3D Augmentation	28.84	20.61	16.38	68.86	48.92	43.57
Geo-CP	23.26	16.41	14.58	60.65	43.93	37.71
Lift3D	22.00	16.61	14.59	63.45	47.34	38.57
RBP	24.92	17.75	15.90	61.99	44.02	38.04
Ours	**29.90**	**21.91**	**16.85**	**69.63**	**49.10**	**43.63**

B.5. Effect of Poisson Blending

We use Poisson blending to enhance the quality of the composition of synthetic cars with the background scene. We observe a slight dip in the detection performance using the obtained augmentations as reported in Tab. 6. A similar observation was made in [20], where improved blending does not positively affect the detection performance.

Table 6. Monocular 3D detection performance of Poisson Blending on our Rendering on KITTI [4] validation set.

(a) MonoDLE[13] on Car with and without Poisson Blending

Rendering	3D@IOU=0.7			3D@IOU=0.5		
	Easy	Mod.	Hard	Easy	Mod	Hard
w/o 3D Aug.	17.45	13.66	11.69	55.41	43.42	37.81
Ours	**22.49**	**15.44**	**12.89**	**63.59**	**45.59**	**40.35**
Ours (+Poisson)	21.34	14.44	12.81	59.60	44.11	38.15

(b) GUPNet[12] on Car with and without Poisson Blending

Rendering	3D@IOU=0.7			3D@IOU=0.5		
	Easy	Mod.	Hard	Easy	Mod	Hard
w/o 3D Aug.	22.76	16.46	13.27	57.62	42.33	37.59
Ours	**23.94**	**17.28**	**14.71**	**61.01**	**47.18**	**41.48**
Ours (+Poisson)	22.43	17.03	14.55	60.00	45.28	39.60

Table 7. Analysis of Training Time

Model	Dataset	Training Time	#GPU's	GPU Model
SA-PlaceNet	KITTI	12h	1	A5000
SA-PlaceNet	NuScenes	32h	1	A5000
GUPNet	Original KITTI	20h	1	A5000
GUPNet	Augmented KITTI	22h	1	A5000
FCOS3D	Original NuScenes	5d18h	2	A5000
FCOS3D	Augmented NuScenes	6d	2	A5000

C. Computational cost of MonoPlace3D

Training of SA-PlaceNet takes a fraction of the time of the detection training. The relative training time is significantly reduced for large datasets such as NuScenes. We present the computational requirements of our augmentation in comparison to the training time in Table 7. We train GUPNet and MonoDLE for an additional 10 epochs and FCOS3D for an additional 5 epochs when training with our augmented data.

Table 8. Data efficiency of SA-PlaceNet on KITTI dataset

MonoDLE		3D@IOU=0.7			3D@IOU=0.5		
% Real Data	% Aug. Data	Easy	Mod.	Hard	Easy	Mod.	Hard
10	10	4.94	3.90	3.26	27.21	21.03	18.06
25	25	13.38	9.78	8.23	48.28	36.99	30.83
50	50	20.46	13.70	11.71	58.04	43.83	37.87
75	75	21.53	14.95	12.38	60.94	45.19	39.99
100	100	**22.49**	**15.44**	**12.89**	**63.59**	**45.59**	**40.35**
100	0	17.45	13.66	11.69	55.41	43.42	37.81

C.1. Data Efficiency on KITTI

In this section, we demonstrate the data efficiency of our method. As observed in Tab.8 our method can significantly reduce the dependence on real data when training monocular detection networks. Specifically, augmenting just 50 % of the real data can achieve better performance than training with 100 % of the original training data.

C.2. Scalability of generated augmentations

To evaluate the effectiveness of the scale of our augmentations, we perform a scalability experiment on a large nuScenes dataset consisting of $\approx 35K$ images. We use different fractions of real and augmented data to train a monocular 3D detector and achieve consistent gains across the amount of data.

Table 9. Scaling on NuScenes dataset

% Data	mAP (w/o aug)	mAP (ours)	NDS (w/o aug)	NDS (ours)
15	0.131	**0.151**	0.223	**0.239**
30	0.231	**0.253**	0.311	**0.339**
50	0.310	**0.342**	0.392	**0.411**
100	0.343	**0.371**	0.415	**0.440**

C.3. Rendering Ablation on NuScenes

We also present an ablation study of various rendering approaches for augmentation in 3D detection for NuScenes. All renderings, when used with our learned placement, outperform the baseline, demonstrating the compatibility of our placement with different rendering methods.

Table 10. Ablation on NuScenes

FCOS3D [1]	MAP	NDS
w/o 3D Augmentation	0.3430	0.415
ShapeNet	0.3441	0.414
Lift3D	0.3460	0.416
Ours	**0.3704**	**0.440**

D. Data Augmentation for Corner Cases

We aim to approximate the training data distribution $p(x)$, with a learned distribution $q_\theta(x)$, which can be sampled ($x \sim q_\theta(x)$) to generate augmentations. In principle, our approach can also model abnormal cases by learning a distribution to approximate the conditional distri-

bution $p(x|state = \text{'abnormal'})$. During inference from SA-PlaceNet we sample the least likely positions from the learned distribution to simulate corners cases for autonomous driving . We augment the training data with these corner cases and train MonoDLE [13] . In Fig 4 we show qualitatively how training with our data can improve the model performance on corner cases .

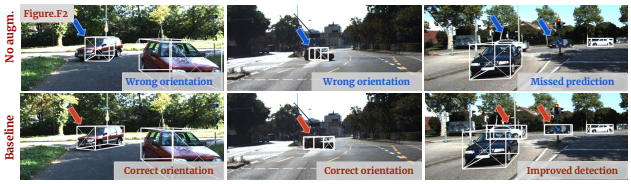

Figure 4. Detection improvement in corner cases.

E. Implementations details

E.1. Placement data Preprocessing

We use the state-of-the-art Image-to-Image Inpainting method [14] to remove vehicles and objects from the KITTI dataset [7]. The input prompt *'inpaint'* is passed to the inpainting pipeline. A few outputs from this method can be seen in Fig. 5

Figure 5. Outputs generated from Stable Diffusion Inpainting pipeline [14]. These inpainted images are used for training our placement model.

E.2. Baseline methods

Geometric Copy-paste (Geo-CP). To augment a given scene, a car is randomly sampled from the database, and its 3D parameters are altered before placement. Specifically, the depth of the box (z coordinate) is randomly sampled, and corresponding x and y are transformed using geometric operations. Other parameters, such as bounding box size and orientation, are kept unchanged. The sampled car is then pasted using simple blending on the background scene.
CARLA [6]. To compare the augmentations generated by simulated road scene environments, we use state-of-the-art CARLA simulator engine for rendering realistic scenes with multiple cars. It can generate diverse traffic scenarios that are implemented programmatically. However, it's extremely challenging for simulators to capture the true diversity from real-world road scenes and they often suffer from a large sim2real gap.
Rule Based Placement (RBP). We create a strong rule-based baseline to show the effectiveness of our learning-based placement. Specifically, we first segment out the road region with [8] and sample placement locations in this region. To get a plausible orientation, we copy the orientation of the closest car in the scene, assuming neighboring cars follow the same orientations. We used our proposed rendering pipeline to generate realistic augmentations.
Lift-3D [10] proposed a generative radiance field network to synthetize realistic 3D cars. Lift3D trains a conditional NeRF on multi-view car images generated by StyleGANs. However, the car shape is changed following the 3D bounding box dimensions. The generated cars are then placed on the road using a heuristic based on road segmentation. We used a single generated 3D car provided in the official code to augment the dataset as the training code is unavailable. Specifically, road region is segmented using off-the-shelf drivable area segmentor [8]. Next, the 3D bounding box of cars is sampled from a predefined distribution of box parameters as given in Tab.11, and the ones outside the drivable area are filtered out. For a sampled 3D bounding box parameters b=$[b_x, b_y, b_z, b_w, b_h, b_l, b_\theta]$, we render the car at adjusted orientation angle $\tilde{\theta}$ using Eq. 1. We place the camera at the fixed height of $1.6m$, with an elevation angle of 0. Also, we used (b_w, b_h, b_l) to render the car of a particular shape. We render the car image for 512x512 resolution using volume rendering and the defined camera parameters. Along with the RGB image, Lift3D also outputs the segmentation mask for the car which is used to blend it with the background. Fig. 6 shows some sample renderings from Lift3D.

Figure 6. Sampled views rendered from Lift3D [10].

F. Rendering details

F.1. Copy-Paste

In simple copy-paste rendering, the cars from the training corpus are added to the predicted 3D bounding boxes. We

Table 11. Preset distribution of bounding boxes. Lift3D [10] samples bounding boxes from the predefined parameter distribution.

Pose	Distribution	Parameters
x	Uniform	$\{[-20m, 20m]\}$
y	Gaussian	$\mu = height, \sigma = 0.2$
z	Uniform	$\{[5m, 45m]\}$
l	Gaussian	$\mu = l_{\text{mean}}, \sigma = 0.5$
w	Gaussian	$\mu = w_{\text{mean}}, \sigma = 0.5$
h	Gaussian	$\mu = h_{\text{mean}}, \sigma = 0.5$
θ	Gaussian	$\mu = \pm\pi/2, \sigma = \pi/2$

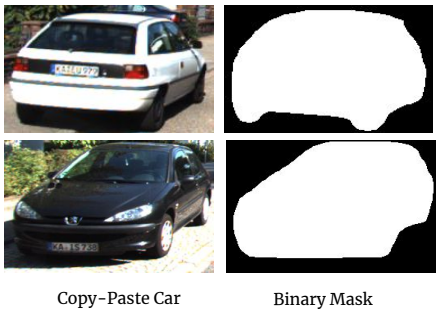

Figure 7. Sample cars from the Copy-Paste Database

extract cars of various orientations from the training set images through instance segmentation using Detectron2 [17]. These cars are archived in a database with their corresponding 3D orientation and binary segmentation mask data. During inference, given a 3D bounding box, we query and search for cars whose orientation closely aligns with the given 3D box orientation. A certain degree of randomness is introduced in selecting the nearest-matching car, contributing to increased diversity and seamless integration with the input scene. Next, we compose the retrieved car image onto the background scene using the 2D-coordinated obtained from the 3D bounding box and the binary mask. This simple rendering essentially captures the diverse cars present in the training dataset and helps in generating scenes that are close to training distribution. However, such rendering has a problem with shadows as the composition is not 3D-aware, given the placed cars are stored as images.

F.2. ShapeNet

ShapeNet [3] is a large-scale synthetic dataset that provides 3D models for various object categories, including cars. The ShapeNet Cars dataset focuses specifically on providing 3D models of different car models from various viewpoints. We leverage the high diversity of cars (nearly 7500 models) in the dataset and render the cars at the predicted box locations with 3D bounding box parameters using Blender [5] software. We employ a random sampling technique to select a 3D car model from this extensive dataset, which is then loaded in the Blender [5] environment. To ensure consistency in the car shapes, we initially calculated the average dimensions of the cars within the dataset. We exclude any car model with dimensions exceeding 50% of the computed average, and we repeat this random sampling procedure until the specified conditions are satisfied. Following that, we align and render the car by a 3D rotation angle. Specifically, as the orientation angle θ is defined in 3D, using it directly to render the image does not take care of perspective projection. Eg. all the cars following a lane will have similar orientation angles (close to zero) but look visually different when projected on the image as shown in Fig. 9. Both the rendered cars have 0 orientation angle in 3D but when projected onto the image planes, the rendered orientation changes with the location. To this end, we adjust the car orientation by a correction factor to incorporate the perspective view, as described in equation (1),

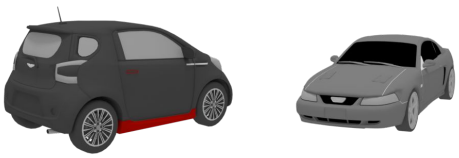

Figure 8. Sample of ShapeNet [3] cars rendered at different views.

$$\tilde{\theta} = \theta + \tan^{-1}(\frac{x}{z}) \tag{1}$$

where x and z are the respective 3D coordinates of the bounding box. We use the final corrected $\tilde{\theta}$ value for rendering the ShapeNet car. We render car images at 512x512, with a white background, which can be later used as a segmentation mask to blend the rendered image. A few examples of the ShapeNet cars rendered with different orientations are visualized in Fig. 8.

Figure 9. Perspective and Absolute projection of cars with the same 3D orientation.

F.3. Reaslistic rendering with Text-to-image models.

We leverage a state-of-the-art image-to-image translation method based on the powerful StableDiffusion model [18] to convert the synthetic ShapeNet renderings into realistic cars. We use edge-conditioned ControlNet [18], which takes an edge image and a text prompt to generate images following the edge map and the prompt. Specifically, we

utilize a canny edge detector to create edge maps for synthetic car images rendered using ShapeNet [3], preserving the car's structure while maintaining its original orientation and scale. These edge maps, generated through the Canny Edge Detection algorithm [2], serve as input for the edge-conditioned ControlNet [18], enabling the rendering of realistic cars using the prompt *'A realistic car on the street'*. Furthermore, given an edge map and hence a ShapeNet-rendered car, we can obtain various realistic renderings at each iteration, facilitating diverse scene generations (Fig. 10). We further enhance ControlNet's backbone diffusion model using LoRA [9] on a subset of 'car' images from the KITTI dataset. This process enables the generation of natural-looking car versions that seamlessly blend with the background scene. Finally, we integrate the ControlNet-rendered car and its shadow base into the predicted location within the scene to achieve a realistic rendering.

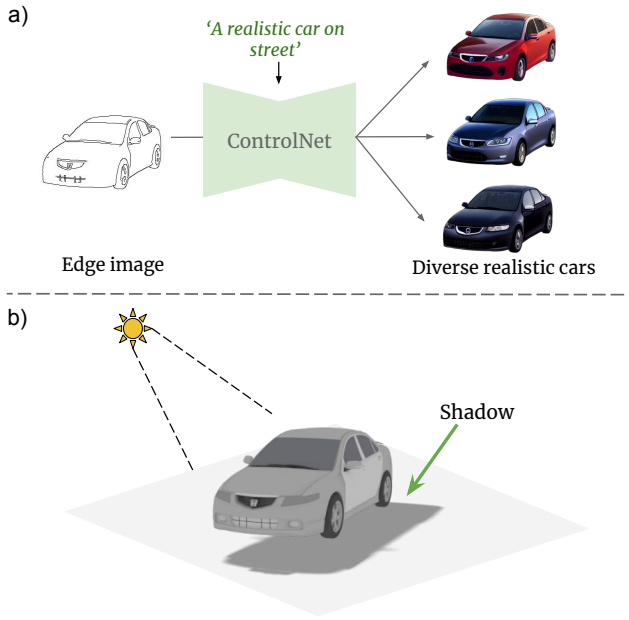

Figure 10. a) Diverse renderings generated with edge-conditioned ControlNet. B) Shadows are generated by rendering 3D assets with a point light source in the blender [5] environment

F.4. Rendering shadows in Blender [5]

To generate a realistic composition of the augmented cars, we generate realistic shadows for cars using the ShapeNet [3] dataset and rendered with Blender. We modify the rendering method to generate shadows by introducing a 2D mesh plane beneath the car base and adding a uniform 'Sun' Light source along the z-axis of the blender environment, placed at the top on the z-axis of the car (Fig. 10). Additionally, we introduce slight variations across all axes for the light source position. Once the cars are positioned within the Blender [5] environment with suitable orientation, we render the entire scene while setting both the car and the 2D plane as transparent. This method enables us to create a collection of shadow renderings with a transparent background for each car in the placement setting.

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *CoRR*, abs/1903.11027, 2019. 1, 3

[2] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. 6

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5, 6

[4] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, page 424–432, Cambridge, MA, USA, 2015. MIT Press. 3

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 1, 5, 6

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 4

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 4

[8] Cheng Han, Qichao Zhao, Shuyi Zhang, Yinzi Chen, Zhenlin Zhang, and Jinwei Yuan. Yolopv2: Better, faster, stronger for panoptic driving perception. *arXiv preprint arXiv:2208.11434*, 2022. 1, 4

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 6

[10] Leheng Li, Qing Lian, Luozhou Wang, Ningning Ma, and Ying-Cong Chen. Lift3d: Synthesize 3d training data by lifting 2d gan to 3d generative radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 332–341, 2023. 4, 5

[11] Hannan Lu, Xiaohe Wu, Shudong Wang, Xiameng Qin, Xinyu Zhang, Junyu Han, Wangmeng Zuo, and Ji Tao. Seeing beyond views: Multi-view driving scene video generation with holistic attention. *arXiv preprint arXiv:2412.03520*, 2024. 2

[12] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3111–3121, 2021. 3

[13] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4721–4730, 2021. 3, 4

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 4

[15] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2397–2406, 2022. 2

[16] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. *CoRR*, abs/1905.12365, 2019. 2

[17] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5

[18] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 5, 6

[19] Renrui Zhang, Han Qiu, Tai Wang, Xuanzhuo Xu, Ziyu Guo, Yu Qiao, Peng Gao, and Hongsheng Li. Monodetr: Depth-guided transformer for monocular 3d object detection. *ICCV 2023*, 2022. 1, 2, 3

[20] Hanqing Zhao, Dianmo Sheng, Jianmin Bao, Dongdong Chen, Dong Chen, Fang Wen, Lu Yuan, Ce Liu, Wenbo Zhou, Qi Chu, Weiming Zhang, and Nenghai Yu. X-paste: Revisiting scalable copy-paste for instance segmentation using clip and stablediffusion. In *International Conference on Machine Learning*, 2023. 3

[21] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *Advances in neural information processing systems*, 27, 2014. 2

[22] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019. 2