ScenDi: 3D-to-2D Scene Diffusion Cascades for Urban Generation

Supplementary Material

In this supplementary document, we first elaborate on our dataset pre-processing process (Sec. 1). Next, we list the details of our network architecture (Sec. 2). Subsequently, we introduce the experimental setup and provide extended qualitative results about the main paper's evaluations, including both 3D generation and 2D image-to-video baselines. Finally, we show our discussions and additional experimental results in Sec. 4 and Sec. 5, respectively.

1. Dataset Preprocess

1.1. Filtering Strategy

Since our method focuses on generating static urban scenes, we eliminate scenarios that exhibit the following three situations:

The first situation is non-ignorable moving objects. We obtain the annotated bounding boxes of each object within the scene and calculate the distance they move to determine whether the objects are dynamic. Next, we project dynamic objects onto each frame and compute the pixels they occupy. If a scene contains visible dynamic objects that occupy an area beyond our predefined threshold, we will filter it out. Filtering thresholds are set as follows: 1-meter movement distance and 1500-pixel minimum occupied area. The rendering resolutions are 640×960 for Waymo and 376×1408 for KITTI-360.

The second situation is the slow ego motion, since the global point cloud can only cover a tiny area. We leverage camera poses from each scene to calculate the translation of ego vehicle. For scenes where the average moving distance between frames is less than 0.25 meters, we consider removing them.

The last situation is the scene with a sharp turn, which can lead to significant differences between the pre- and post-curve environments, making it challenging to obtain reliable 3D point clouds. We avoid this by fitting the camera position for each frame into a straight line and omitting scenes with notable directional deviations.

2. Network Architecture

2.1. Voxel-to-3DGS VQ-VAE

Due to the fact that 3D convolutional layers significantly consume GPU memory, we employ Asymmetrical Residual Blocks from Cylinder3D [22] as the base convolutional block to construct our VQ-VAE architecture. We set 128 as our base feature channels. The encoder first uses a 3D convolution layer to transform the input channel to 128, then it goes through 3 downsampling blocks which increase the feature channels to 128, 256, 256, respectively. Each downsampling block contains 2 base convolutional blocks and a downsampler with a stride 2 despite the last block. A middle block is employed in the bottleneck, which contains 2 base convolutional blocks and 1 attention block. Another 3D convolution layer reduces the feature channel to 8 in the latent space. We adopt codebook size $\mathcal{C} = 8192$ on both datasets, and employ EMA vector quantizer [11] for stable training. After quantization, the decoder first uses a 3D convolution layer to increase channels to 256, followed by another middle block to augment information in lower dimensional space. Then 3 upsampling blocks each contains 2 basic convolutional blocks and an upsampler with a stride 2 (implemented as transpose convolution layer) will decrease the feature channels to 128. Finally, a DDCM block also borrowed from Cylinder3D with an occupancy head and a feature head will predict the coarse 3D geometry occupancy value and its corresponding feature within each voxel. Each attribute decoder is a 4-layer MLP with an intermediate feature dimension 128. Hyperparameters for VQ-VAE are listed in Tab. 1.

Hyperparameter	3D VQ-VAE
Base Block Channel	128
Channel Multiple	$1, 2, 2$
Latent Dim	8
Codebook Size	8192
Embedding Dim	8
Scaling Factor	4
Feature Dim	16
MLP Layer	4
MLP Layer Width	128

Table 1. Hyperparameters of 3D VQ-VAE

Hyperparameter	3D Diffusion
Diffusion Steps	1000
Inference Steps	100
Noise Schedule	cosine
Input Channel(unconditional)	8
Patch Size	2
Hidden Size	768
Num Heads	16
MLP Ratio	4.0
Num of Blocks	24

Table 2. Hyperparameters of 3D DiT

2.2. 3D Diffusion

The 3D Diffusion model is a 3D DiT inherited from the architecture of [9, 10]. The hyperparameters of our diffusion model are listed in Tab. 2.

3. Experiment Details and Qualitative Results

3.1. 3D Generation Baselines

3.1.1. Camera Controllability Assessment

To evaluate camera controllability of 3D baselines and our method, we set up identical camera trajectory configurations for all approaches. Specifically, we designed two types of trajectories: rotation and translation. The rotation trajectories are randomly sampled from left/right turns between 15 to 60 degrees, and the translation trajectories are randomly sampled from left/right shifts between 2 to 3 meters. Qualitative results for all methods under customized rotation and translation trajectories are shown in Fig. 1. Since all methods are trained exclusively on forward-facing camera data, 3D baselines like CC3D [1] and Discoscene [19] struggle to perform well under large camera rotation. When we apply larger rotation angles, the image quality of these baselines significantly degrades, leading to failures in scene reconstruction, and hence Dust3R [17] fails to reconstruct most scenes under such conditions. Across all test trajectories, our method demonstrates camera control capabilities comparable to pure 3D methods, while producing higher-quality images, which in turn results in better camera pose estimation results.

3.1.2. Qualitative Results on Waymo

As our model is also trained on Waymo [15] dataset, we provide qualitative results on our method and all 3D baselines when performing forward driving on Waymo in Fig. 2.

3.2. Image-to-Video Generation Baselines

3.2.1. Vista Trajectory Modeling

In contrast to other approaches, the camera in Vista [3] is governed by high-level commands (e.g., speed+steering angle, 2D trajectory) , which prevents us from adopting fully identical trajectories used for other baselines. We adopt one of the control signals recommended by the authors—specifically, speed and steering angle—for our camera controllability evaluation. In this section, we elaborate on the modeling details for transforming speed and steering angle into a camera trajectory.

Kinetic Model: We use a simplified bicycle model to describe the motion. As shown in Fig. 3. The vehicle moves along a circular path with a fixed radius (R), determined by the steering angle and wheelbase:

$$R = \frac{L}{\tan(\theta_k)}. \tag{1}$$

The angular velocity ω:

$$\omega = \frac{v}{R}. \tag{2}$$

The angular increment per frame $\Delta\theta$(Δt is the time interval per frame):

$$\Delta\theta = \omega \times \Delta t. \tag{3}$$

Position and Orientation Calculation: In a world coordinate system where +x is left, +y is upward, and +z is forward, the vehicle's initial position (at frame k=0) is (0, 0), and its orientation $\theta = 0$ (along the positive z-axis). The vehicle moves along a counterclockwise circular arc with radius R , with the circle's center located to the left or right of the initial position, i.e., at $(0, -R)$ or $(0, R)$. For the (k)-th frame, the time is $t_k = k\Delta t$ seconds. Orientation angle:

$$\theta_k = \omega t_k. \tag{4}$$

Position (starting from ((0, 0)), adjusted for the circle's center):

$$x_k = R(1 - \cos(\theta_k)), \tag{5}$$

$$z_k = R\sin(\theta_k). \tag{6}$$

Camera Pose Representation: We assume the camera aligns with the vehicle's coordinate system, where the z-axis follows the vehicle's heading, the x-axis points left, and the y-axis points upward. For the (k)-th frame, the camera is located at $(x_k, 0, z_k)$ with an orientation angle θ_k. The transformation matrix T_k:

$$T_k = \begin{bmatrix} \cos(\theta_k) & 0 & -\sin(\theta_k) & x_k \\ 0 & 1 & 0 & 0 \\ \sin(\theta_k) & 0 & \cos(\theta_k) & z_k \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

Calculation Example: Using an example with a vehicle speed of 10.8 km/h and a steering wheel turned 780 degrees to the left. The known conditions are as follows:

- Time interval per frame Δt: the official sampling frequency of Vista is $10Hz$. The time interval per frame is $\Delta t = 0.1$ seconds.
- Vehicle speed: 10.8 km/h, converted to meters per second: $10.8 \times \frac{1000}{3600} = 3$ m/s.
- Steering wheel angle: Assuming a steering ratio of 20:1, the wheel steering angle is $780/20 = 39.5°$.
- Number of frames: 25 frames, from frame 0 to frame 24, corresponding to time $t = 0$ seconds to $t = 2.4$ seconds. The first frame ($t = 0$) is taken as the origin of the world coordinate system, with the initial orientation along the positive z-axis.
- Assumptions: We assume $L = 2.5$m (a common car wheelbase). The camera is mounted at the vehicle's center and aligned with the vehicle's orientation.

The pose for each frame can now be calculated by substituting $k = 0$ to 24 into the above equations for θ_k, x_k, and y_k, and constructing the corresponding T_k matrices.

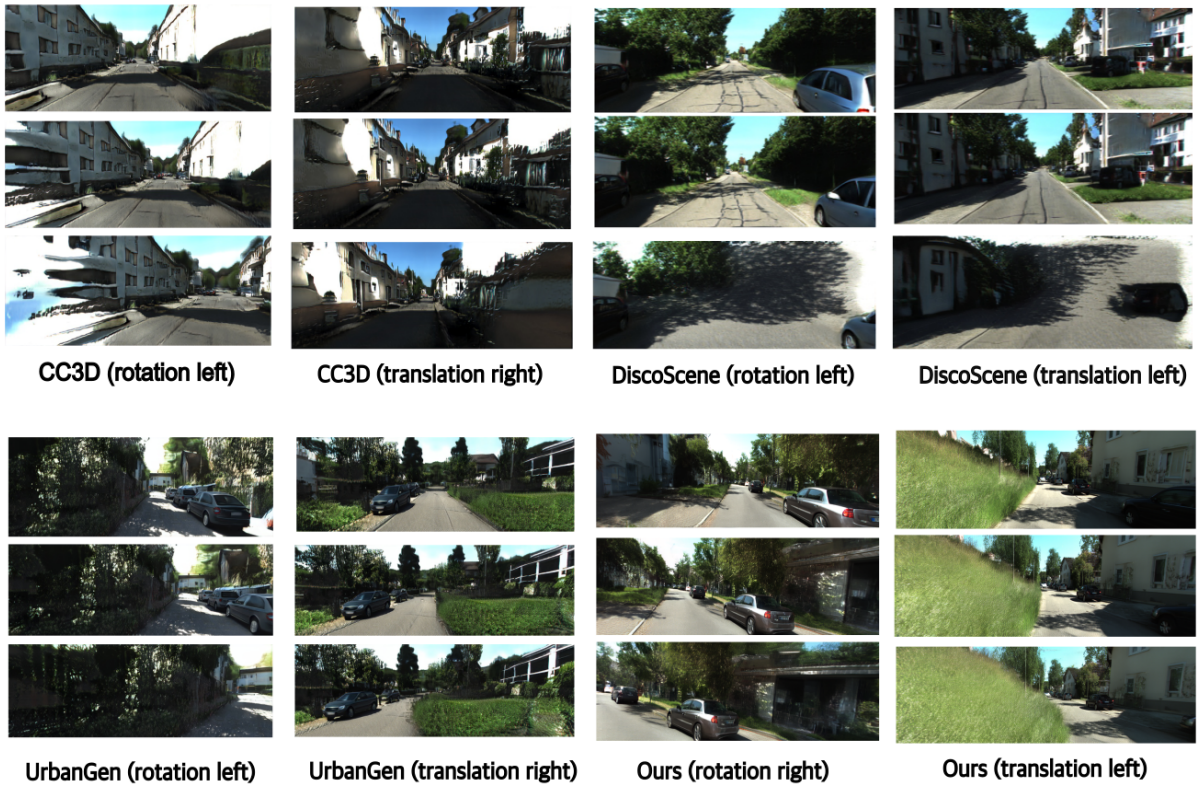

Figure 1. **Qualitative results of camera controllability evaluation on 3D generation baselines.** We show corresponding results of different methods under different camera control.

3.2.2. Camera Controllability Assessment

For Gen3C [13], we choose the same trajectories as other methods. For Vista [3], we adopt two different steering angle settings: a normal turn with 156°, and a maximum steering angle of 780°, both at a consistent speed of 10.8 km/h. Qualitative results of both methods are shown in Fig. 8. Benefiting from its 3D cache, Gen3C successfully adheres to the given camera trajectory. In contrast, we observe that the pure 2D-based approach, Vista, tends to ignore the steering angles and maintains a constant orientation, only demonstrating effective directional control at intersections. Increasing the steering angle to 780° results in minimal orientation changes, and image quality often degrades.

3.2.3. More Qualitative Results of Gen3C

When Gen3C is applied to forward-driving scenarios, we observe the following issues. First, long-distance forward driving causes 3D cache rendering failures, leading to obvious artifacts characterized by sudden changes in color and content in generated videos. Second, when the overlap between input image and the target generation region is small, such as during substantial camera translation, we observe abrupt scene inconsistencies. Both cases result in significant quality degradation in the generated video. Since FID and FVD are highly sensitive to anomalous outliers, these severe degradations cause the scores to increase. This explains why the FID and FVD results are not outstanding, while the KID metric, which is less sensitive to such anomalies, yields favorable results. The qualitative results of the aforementioned situations are shown in Fig. 4.

4. Discussion

4.1. 3D Backbone Choice

For our Voxel-to-3DGS VQ-VAE, we use Asymmetrical Residual Blocks (ARB) from Cylinder3D [22] that decomposes the 3D convolution into 2D convolution and 1D convolution, which is more memory efficient compared with the plain 3D convolution. We also attempt to use Sparse Convolution as a potential backbone. Specifically, we inherit the VAE of XCube [12] as it already encodes both semantic and geometry information about urban scenes. We conduct generalizable reconstruction experiments on 50

CC3D [1] DiscoScene [19] UrbanGen [20] Ours

Figure 2. **Qualitative Comparison on Waymo** with cameras moving forward.

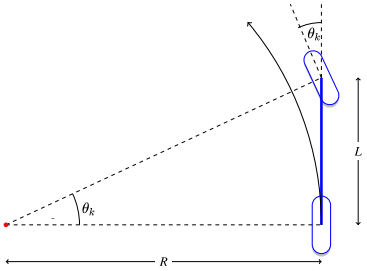

Figure 3. **Bicycle model.** Schematic diagram of the bicycle model [16].

Figure 4. **Qualitative results of Gen3C** with cameras moving forward. Row 1 (highlighted by the yellow box) shows visual artifacts resulting from depth accumulation error. Row 2 demonstrates the abrupt degradation in scene quality, and Row 3 provides a successful example of forward scene generation.

scenes from KITTI-360 [7] dataset and show the test view reconstruction results in Fig. 6. Interestingly, our experiments reveal that under the same voxel size (both at 0.4 in our settings), the ARB-based implementation outperforms sparse representations in terms of reconstruction quality (col. 2 vs col. 3). To further explore this phenomenon, we increase the resolution of input sparse voxel grids (reducing voxel size to 0.2), and observe higher performance. However, even with higher resolution, its reconstruction quality does not show clear improvement compared to the ARB-based variant (col. 2 vs col. 4).

While the sparse convolution is more memory efficient to enable higher input voxel resolution, this advantage comes at the cost of higher latent dimension, which imposes greater challenges for the diffusion model to model plausible latent distributions. As shown in prior works (e.g., [18]), training a diffusion model on high-dimensional input often leads to worse performance. Considering the trade-off between memory cost and latent dimension, we choose the ARB-based convolution for our final reconstruction backbone.

4.2. Depth Estimator Choice

The quality of the input voxel grid depends on the accuracy of the monocular depth estimator used for point cloud construction. To investigate whether our VQ-VAE is sen-

sitive to this choice, we compare two depth estimators: our default Metric3D [4] and MapAnything [5], a recent model following the emerging feed-forward paradigm of directly predicting 3D-aware outputs from images, which can produce more geometrically accurate depth estimates. For a fair comparison, both variants are trained on the same 500 scenes from KITTI-360 and evaluated on held-out test views.

As shown in Tab. 3, MapAnything achieves consistently better PSNR, SSIM, and LPIPS scores compared to Metric3D. This suggests that while our method can work reliably with off-the-shelf depth estimators, it is not limited by any particular choice and can naturally benefit from stronger depth estimation as the field advances.

	PSNR↑	SSIM↑	LPIPS↓
Metric3D [4]	24.96	0.82	0.25
MapAnything [5] (w. pose)	25.91	0.85	0.19

Table 3. VQ-VAE reconstruction quality using different depth estimators, trained and evaluated on the same set of scenes.

4.3. Depth Error Propagation

A natural concern is whether errors in the estimated depth maps propagate through the pipeline and degrade the final generation quality. We argue that our design mitigates this issue at multiple levels.

First, the VQ-VAE is supervised by ground truth RGB images rather than by the noisy voxel grid itself. This means the model implicitly learns to recover plausible geometry and appearance even from imperfect inputs. Second, each Gaussian is equipped with a learnable positional offset, which allows the model to correct local geometric errors induced by inaccurate depth estimates. Finally, since the 3D generation stage only needs to produce a coarse structural prior to guide the 2D diffusion model, high-precision depth is not a critical bottleneck for final rendering quality.

To empirically verify this, we conduct a toy experiment: instead of running the full generation pipeline, we directly feed the VQ-VAE reconstruction into the 2D diffusion model. This allows us to trace depth information at each stage — from the raw Metric3D depth, to the rendered depth from the VQ-VAE output, to the final synthesized image — as visualized in Fig. 5. Despite the noisy Metric3D depth and the smoothed geometry in the VQ-VAE reconstruction, the foreground regions in the final output closely resemble the original scene. Background and sky regions, which are entirely handled by the 2D diffusion model, are unaffected by depth estimation quality. Together, these results confirm that depth errors do not critically propagate to the final synthesis.

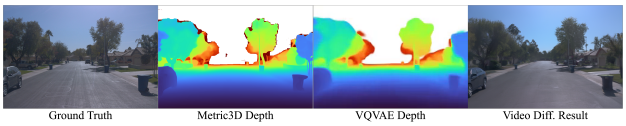

Figure 5. **Visualization of Depth Propogation**

4.4. Why Combining 3D & 2D Diffusion Models

Works like L3DG [14] demonstrate that it is possible to directly predict 3D Gaussians via 3D latent diffusion. However, their approach requires time-consuming per-scene optimization to prepare 3DGS parameters for VQ-VAE training. In this work, we investigate a more efficient alternative that bypasses this bottleneck by using monocular-depth-based point clouds to form the input voxel grid, and learn a Voxel-to-3DGS VQ-VAE. While missing Gaussian attributes such as scale and rotation hinder direct 3DGS decoding, we overcome this by fusing coarse 3D information with 2D video diffusion priors, leveraging their complementary strengths for high-fidelity synthesis.

5. More Experimental Results

5.1. Inference Time

We report the inference time of our method broken down by each stage, as well as a comparison with image-to-video baselines, in Tab. 4. Our 3D diffusion and VQ-VAE decoding stages are highly efficient, taking only 5s and 0.25s for 81 frames. The overall inference time is dominated by the 2D video diffusion backbone, which is shared with competing I2V methods. This indicates that future improvements in video diffusion efficiency will directly translate to faster inference for our method.

	Ours			Vista	Gen3C
	3D diff.	VQVAE dec.	2D diff.		
Speed (min/frames num.)	0.08/81	0.004/81	4.8/81	6.8/100	9.7/121

Table 4. Inference Time Comparison with Baselines on A100

5.2. Forward-Backward Test

Compared with methods [2, 6, 8, 21] using only 3D geometry buffers for video generation, we do a Forward-Backward Test to show our advantages over them in preserving consistency. Since those methods generate scenes in an autoregressive manner, it means that they can not memorize previous contents when they trace back to the original location. Our method instead always provides a coarse appearance signal to remind model of the corresponding scene information. The result can be seen in Fig. 7. Both scenes use the same camera trajectory to move forward and backward the same distance. We observe significant changes in scene contents generated by the depth map condition variant, while our model generates scenes more consistent with

Ground Truth | ARB Conv (0.4) | Sparse Conv (0.4) | Sparse Conv (0.2)

Figure 6. **Qualitative Comparison of different backbone on KITTI-360 dataset**. The number in brackets indicates the voxel size.

the original one, emphasizing the importance of 3D representation in loop consistency.

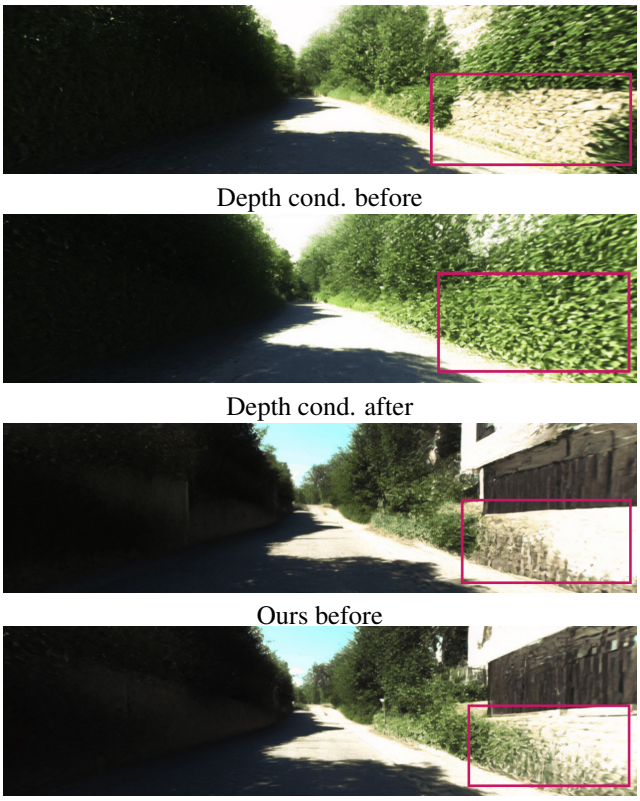

Figure 7. **Forward-Backward Test Results on KITTI-360.**

5.3. Random Samples

We show more randomly sampled results on both KITTI-360 and Waymo dataset in Fig. 9 and Fig. 10.

References

[1] Sherwin Bahmani, Jeong Joon Park, Despoina Paschalidou, Xingguang Yan, Gordon Wetzstein, Leonidas Guibas, and Andrea Tagliasacchi. Cc3d: Layout-conditioned generation of compositional 3d scenes, 2023. 2, 4

[2] Boyang Deng, Richard Tucker, Zhengqi Li, Leonidas Guibas, Noah Snavely, and Gordon Wetzstein. Streetscapes: Large-scale consistent street view generation using autoregressive video diffusion. In *SIGGRAPH 2024 Conference Papers*, 2024. 5

[3] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 2, 3, 7

[4] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation, 2024. 5

[5] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*. IEEE, 2026. 5

[6] Bohan Li, Jiazhe Guo, Hongsi Liu, Yingshuang Zou, Yikang Ding, Xiwu Chen, Hu Zhu, Feiyang Tan, Chi Zhang, Tiancai Wang, et al. Uniscene: Unified occupancy-centric driving scene generation. *arXiv preprint arXiv:2412.05435*, 2024. 5

[7] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv preprint arXiv:2109.13410*, 2021. 4

[8] Yifan Lu, Xuanchi Ren, Jiawei Yang, Tianchang Shen, Zhangjie Wu, Jun Gao, Yue Wang, Siheng Chen, Mike Chen, Sanja Fidler, and Jiahui Huang. Infinicube: Unbounded and controllable dynamic 3d driving scene generation with world-guided video models, 2024. 5

[9] Shentong Mo, Enze Xie, Ruihang Chu, Lewei Yao, Lanqing Hong, Matthias Nießner, and Zhenguo Li. Dit-3d: Exploring plain diffusion transformers for 3d shape generation. *arXiv preprint arXiv: 2307.01831*, 2023. 1

[10] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022. 1

Figure 8. **Qualitative results of camera controllability evaluation on image-to-video baselines.** Vista [3] demonstrates effective directional control at intersections (row 1); however, maintaining precise direction near straight sections poses a challenge (row 2,3). Conditioned on a 3D point cloud, Gen3C [13] successfully adheres to the given trajectories (row 4,5).

[11] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. 1

[12] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 3

[13] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 3, 7

[14] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, Angela Dai, and Matthias Nießner. L3dg: Latent 3d gaussian diffusion. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. 5

[15] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[16] Mario Theers and MankaranSingh. thomasfermi/algorithms-for-automated-driving: zenodo doi release, 2023. 4

[17] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 2

[18] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, and Pan Ji. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. *ACM Transactions on Graphics*, 43(4), 2024. 4

[19] Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Skorokhodov Ivan, Siarohin Aliaksandr, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, and Tulyakov Sergy. Dis-

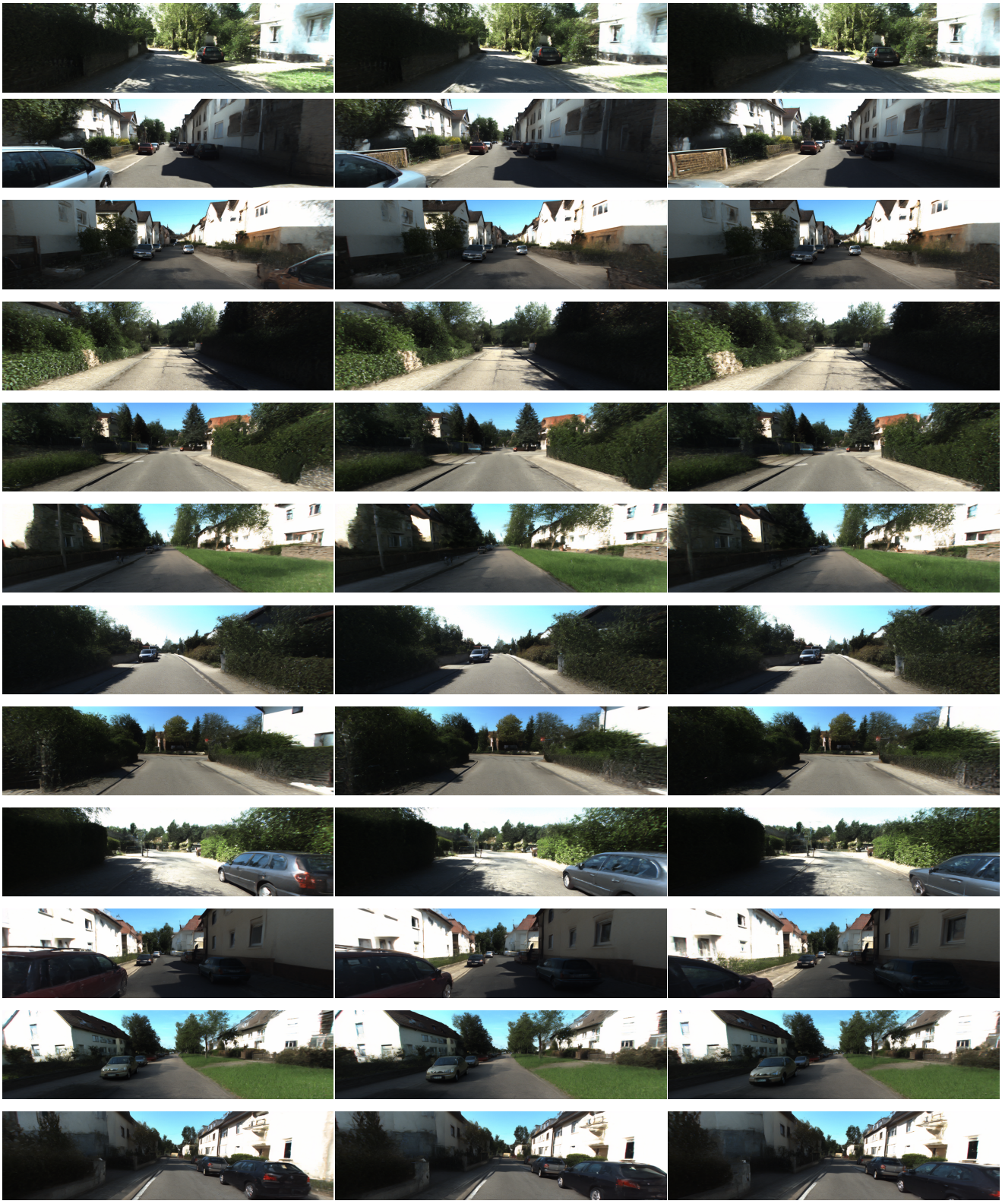

Figure 9. **More samples on KITTI-360.**

Figure 10. **More samples on Waymo.**

coscene: Spatially disentangled generative radiance field for controllable 3d-aware scene synthesis. *arxiv: 2212.11984*, 2022. 2, 4

[20] Yuanbo Yang, Yujun Shen, Yue Wang, Andreas Geiger, and Yiyi Liao. Urbangen: Urban generation with compositional and controllable neural fields. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence, pages 1–17, 2025. 4

[21] Benjin Zhu, Xiaogang Wang, and Hongsheng Li. Consistentcity: Semantic flow-guided occupancy dit for temporally consistent driving scene synthesis. In *IEEE/CVF International Conference on Computer Vision*, 2025. 5

[22] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. *arXiv preprint arXiv:2011.10033*, 2020. 1, 3