Supplementary Material for Outdoor Scene Extrapolation with Hierarchical Generative Cellular Automata

A. Additional Analysis

In this section, we include additional results that highlight the practical aspects of the proposed method.

A.1. Evaluation on Real-World Data

A.1.1 Waymo-Open

We first provide further evaluations on the sim-to-real performance of hGCA using Waymo-Open [17] dataset. As stated in the main manuscript, abundant real-world AV data suffers from various noises and limited measurement ranges. Most importantly, we do not have the ground truth shapes for our task of shape extrapolation, which challenges systematic analysis. This section provides sub-optimal quantitative measures and comprehensive qualitative results to demonstrate that we can faithfully generate realistic scenes given partial and noisy real-world measurements.

As a means for quantitative evaluation, we use the accumulated scans as a pseudo ground truth as in semantic scene completion works [1, 20, 21] and analyze its performance with LiDAR ReSim and IoU as demonstrated in our results of synthetic datasets. We randomly chose 202 scenes and used five scans as input. Then, the generated scene is compared against accumulated data using all scans. While the accumulated scans are denser variations of the given measurement, they are noisy and highly incomplete measurements in a confined height range, as shown in Fig. 1, 2, 3 and 4. We devise the evaluation metrics to adapt to the limitation of the reference data. When computing the LiDAR ReSim score, we simulate LiDAR at the same height as the *input* LiDAR scan instead of using a higher elevation LiDAR. Likewise, we compute IoU only on regions visible from LiDAR scans as evaluated in semantic scene completion [1, 20, 21]. Therefore, neither of these metrics assesses the performance of extrapolation beyond the LiDAR height range and occlusion.

Table 1 reports the performance on the shape completion within the measurement range, only trained with limited synthetic content. Similar to synthetic results, hGCA outperforms all baselines in LiDAR ReSim and IoU by a margin. The results indicate that our completion is closer to the dense measurements of real-world geometry than existing methods. Interestingly, the IoU of the continuous completion outperforms the initial voxel occupancy (10cm) in the real-world analysis in Table 1 whereas the voxel occupancy achieves higher IoU values in our synthetic experiments (Table 1 of the main paper). Recall that our high-resolution upsampling sometimes fails to create a narrow structure, as unsigned distance fields may obscure the exact zero-level location in high-frequency details. We observe that the accumulated scans also experience similar ambiguity due to the inherent noise in the real-world measurement. As the noisy scans serve as the ground truth, the possible misalignment due to the thick implicit generation may be evaluated as a faithful generation. Such a phenomenon again shows difficulties in the quantitative evaluation of generative models on real-world datasets.

Method	Represe -ntation	LiDAR ReSim min. ↓	LiDAR ReSim avg. ↓	LiDAR ReSim TMD ↑	IoU
SCPNet	20cm	5.57		-	52.33
SG-NN	10cm	5.81		-	49.83
GCA	20cm	5.73	6.04	**1.07**	51.91
GCA + Planner	20cm	5.48	5.57	0.70	52.26
hGCA	10cm	4.65	4.73	0.77	52.25
	implicit	**4.52**	4.50	0.97	**56.50**

Table 1. Quantitative results on Waymo with 5 scans given as input. All results except IoU are multiplied by 10 in meter scale. LiDAR Resim evaluates the fidelity of completion and TMD measures the diversity of generation. Unlike synthetic results, LiDAR ReSim uses same elevation angle as the input and IoU is computed with accumulated scans.

We further visualize various generation results observed from diverse viewpoints in Fig. 1, 2, 3 and 4. Despite the limited measurement range and noisy input, our method can extrapolate the input measurement into large-scale real-world scenes in a scalable way. Figure 2, 3 and 4 also show comparison against other baselines. Deterministic baselines (SCPNet [20] and SG-NN [5]) exhibit conservative behavior, leaving holes in the ground and partially complete buildings or trees. Naïve GCA, while it is a generative baseline,

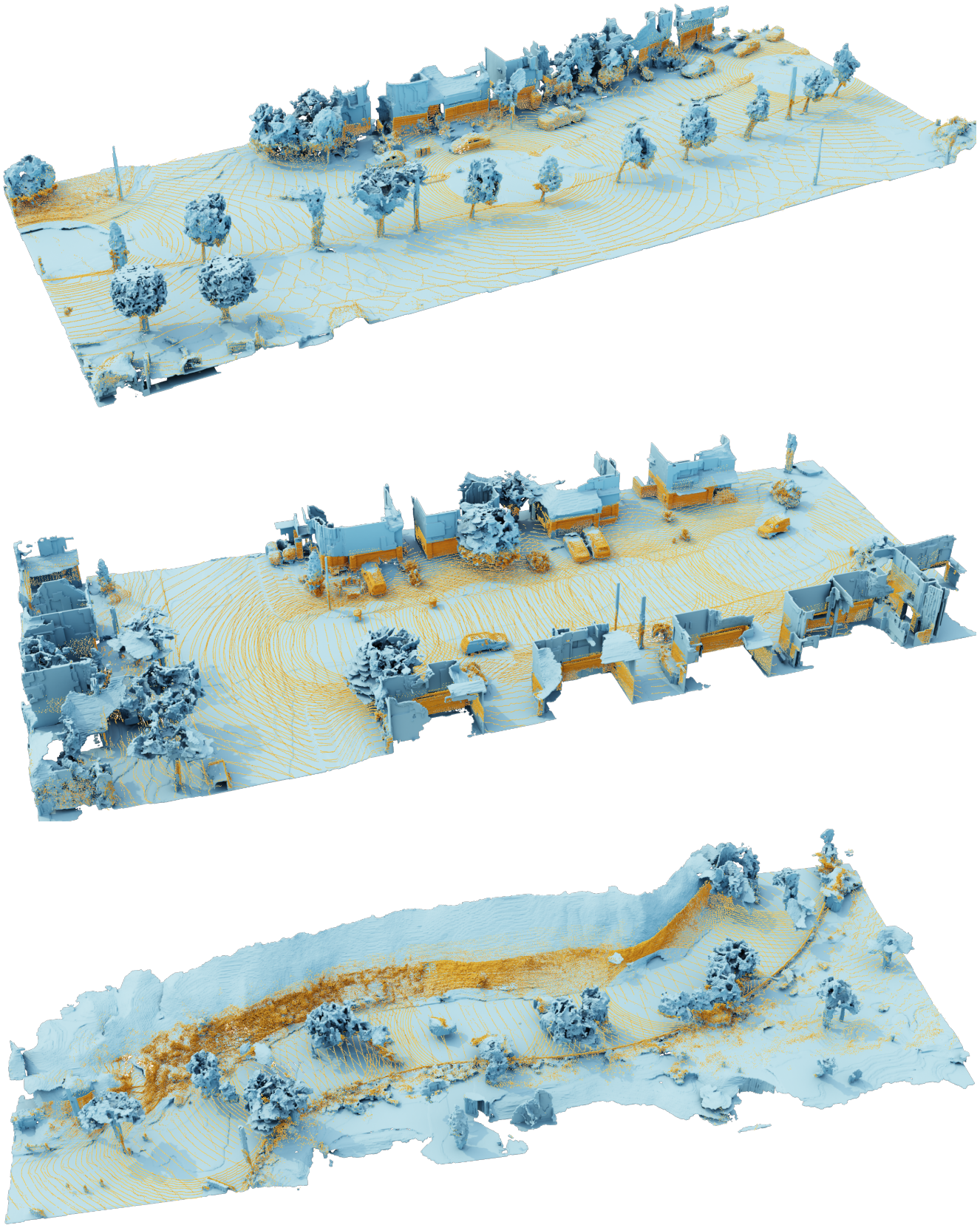


Figure 1. Additional completion visualizations on real-world Waymo-Open dataset on 100m scenes. Yellow spheres indicate input. hGCA is spatially scalable, completing this whole scene (100 meters) at high resolution on a single 24GB GPU without additional tricks. hGCA can even extrapolate hills (bottom) from real-world scans.

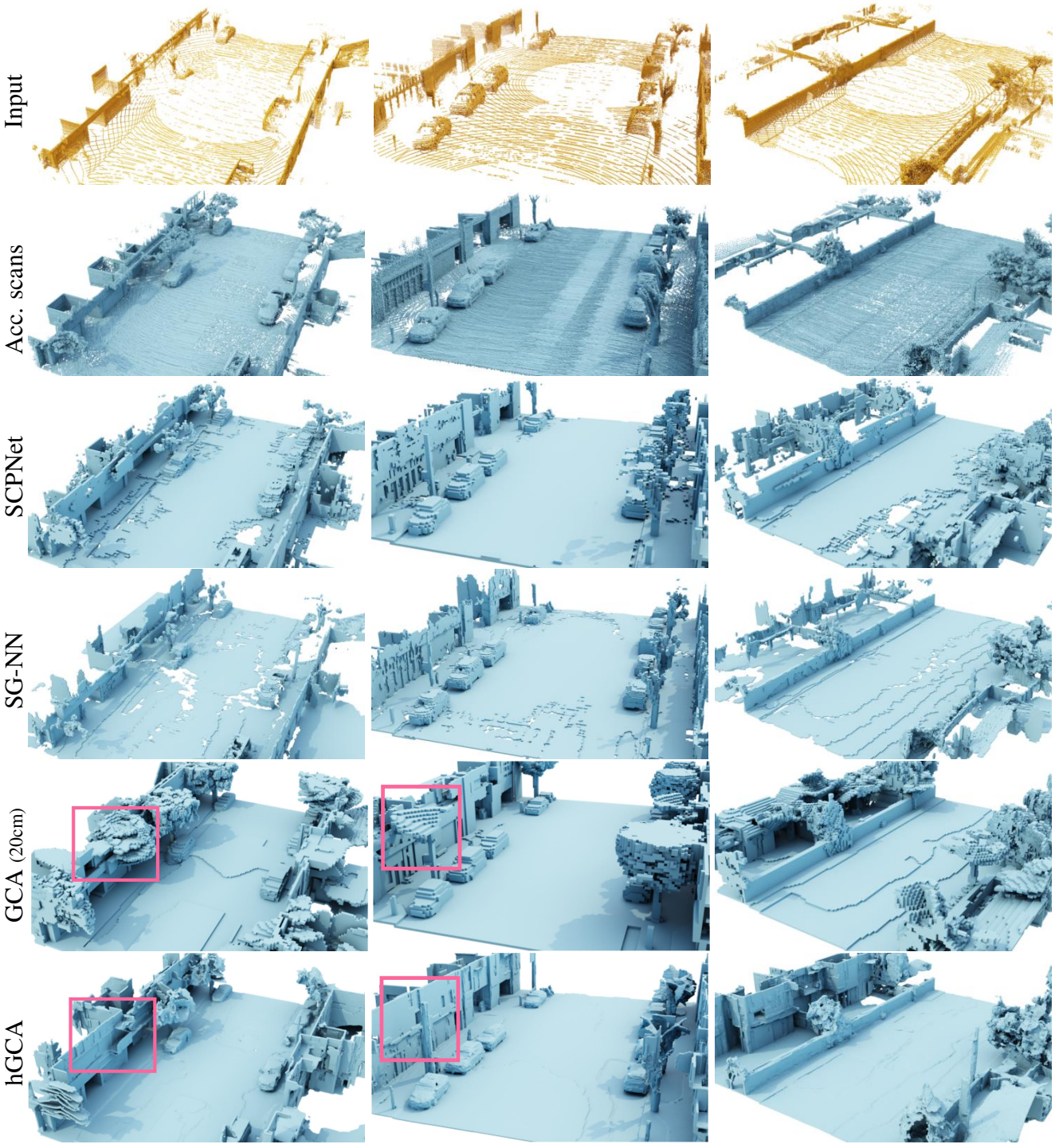


Figure 2. Additional visualizations on real-world Waymo-open dataset. hGCA exhibits great sim-to-real performance, while naive GCA suffers from inconsistency (pink).

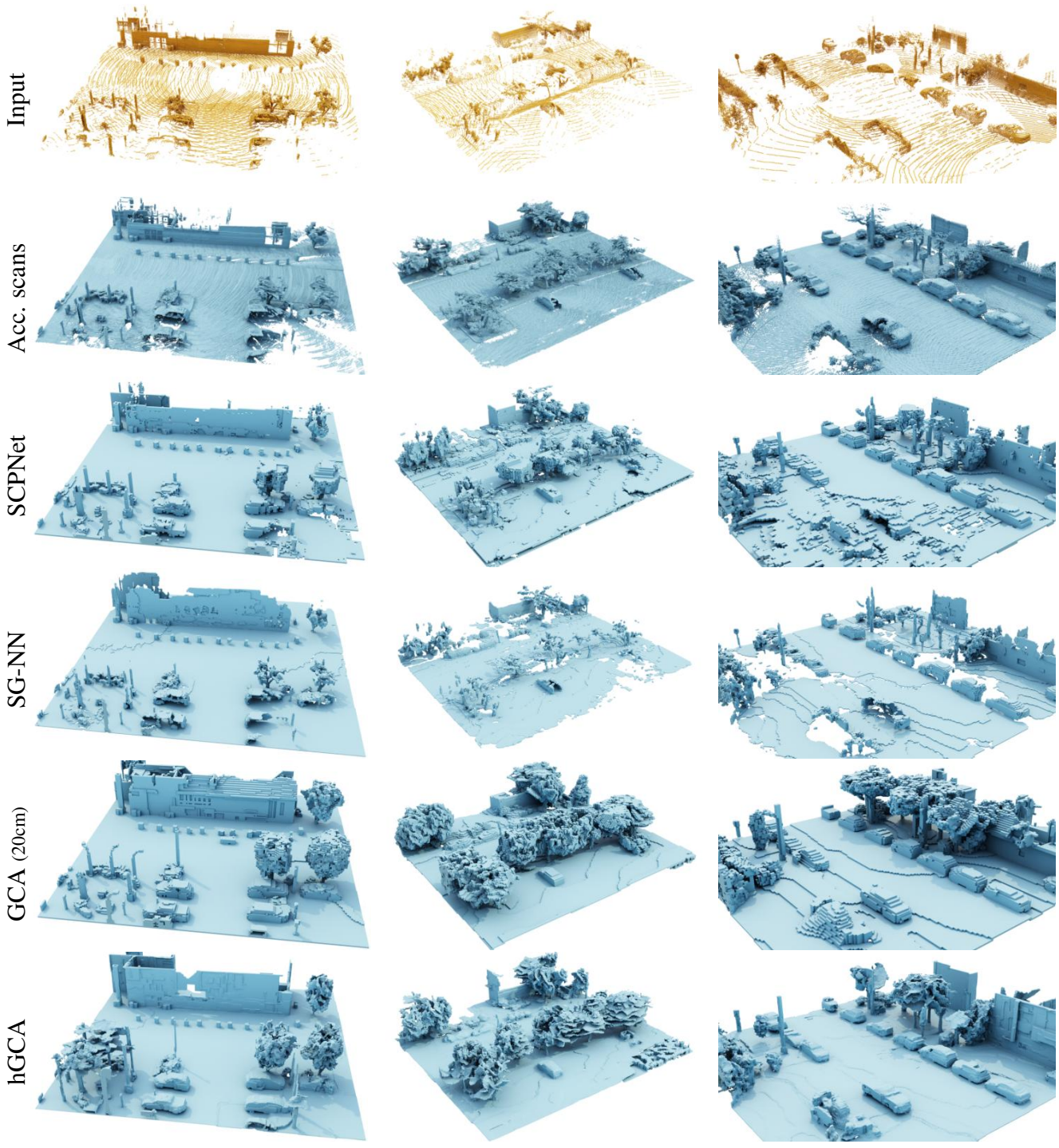


Figure 3. Additional visualizations on real-world Waymo-open dataset.

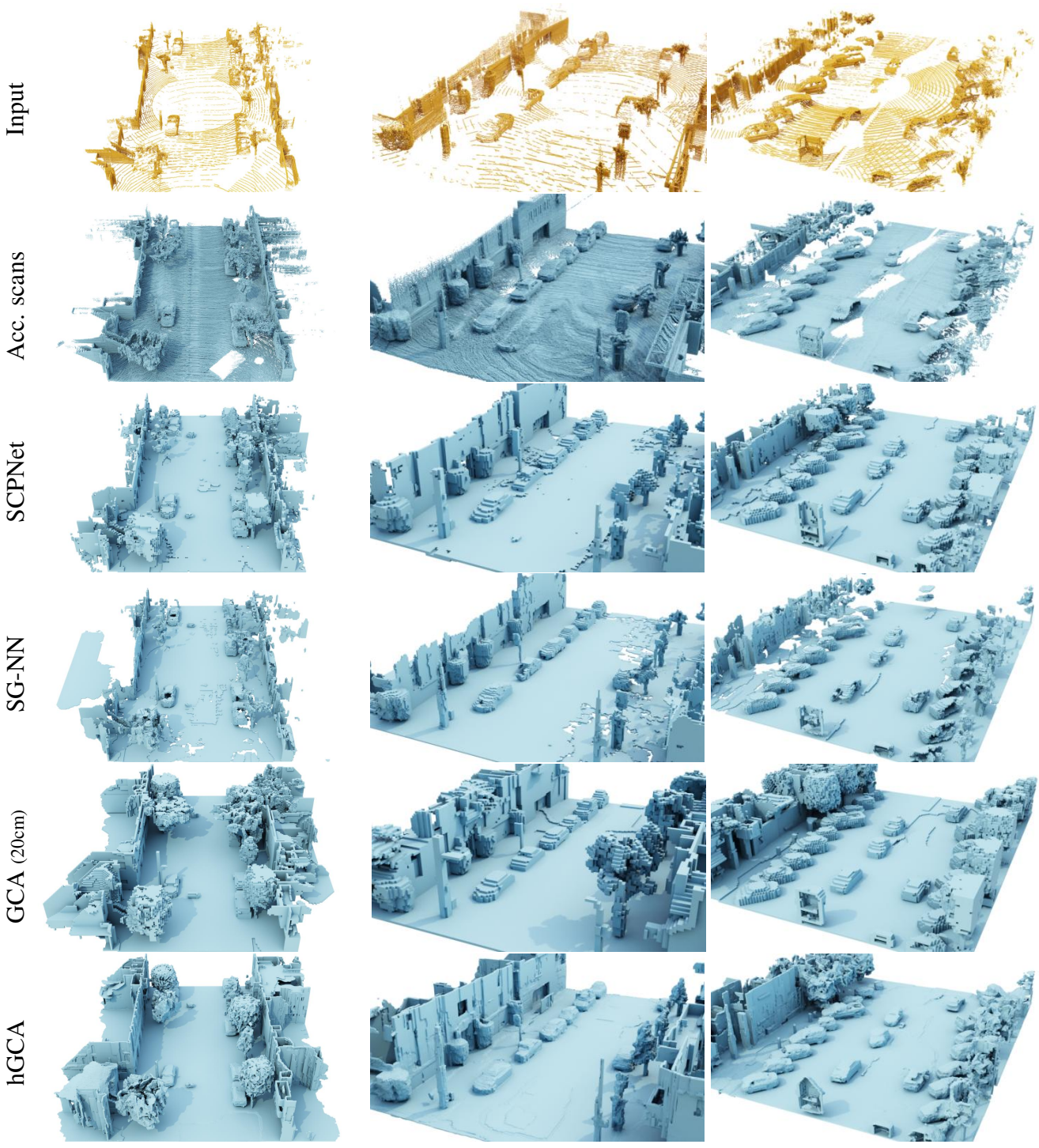


Figure 4. Additional visualizations on real-world Waymo-open dataset.

	Sparse Scene				Sparse Car			
Method	5%	10%	50%	100%	5%	10%	50%	100%
SCPNet (20cm^3)	10.44	2.33	2.28	2.25	2.38	2.30	2.37	2.25
JS3CNet (20cm^3)	96.59	112.65	170.16	2.40	2.59	2.52	2.43	2.4
SG-NN (10cm^3)	133.61	133.28	184.4	1.96	2.45	2.19	2.03	1.96
GCA (20cm^3)	2.47	2.27	2.44	2.27	2.38	2.51	2.38	2.27
hGCA (10cm^3)	**2.41**	**2.11**	**1.77**	**1.71**	**2.30**	**2.08**	**1.82**	**1.69**

Table 2. Chamfer distance between the ground truth geometry and completions by varying sparsity. Sparse scene and sparse car indicates a scenario where we sparsify the regions of entire scene (including ground) and only the car, respectively. Chamfer distance above ground are reported and we report average distance of $k = 3$ generations for generative models (GCA, hGCA). hGCA generalizes well to sparse, novel data.

suffers from inconsistent generation, which results in overlapping structures (pink boxes in Fig. 2). On the other hand, hGCA can generate unseen geometry while maintaining global consistency in various real-world settings. For example, the input shown in the right column of Fig. 2 misses a significant portion of the buildings as a fence occludes them. Nonetheless, hGCA faithfully generates detailed facades of buildings.

A.1.2 nuScenes

To test generalization to a new sensor configuration, we show results on the nuScenes dataset in Fig. 5. While 64-beam LiDAR was used in Waymo and our synthetic training set, nuScenes scans were obtained using 32-beam LiDAR, which is a significant domain shift. Nevertheless, hGCA generalizes to nuScenes out-of-the-box and is robust to the domain shift. We observe some failures on nuScenes (bottom of Fig. 5), across the street, where the model hallucinates structures in extremely sparsely scanned regions. We speculate that training with simulated 32-beam LiDAR scans as input will enhance the quality of completion by reducing the domain gap for sparse inputs.

A.2. Generalization to Various Input Conditions

As partially demonstrated in sim-to-real results, our generative pipeline can robustly handle input variations not observed in the training data. This section provides additional quantitative evaluations of shapes generated from different input conditions.

Input Sparsity. We first demonstrate the performance of hGCA given inputs of varying densities of the entire scene. We place a three-wheeler[1], verified to be unseen from training data, in the center on a flat ground and simulate LiDAR scans captured from a simple trajectory. Then, we vary the input density by randomly sampling 5%, 10%, and 50% of the accumulated scans of two settings: the whole scene and only the car.

We report Chamfer distance between the completion and the ground truth in Table 2 and visualize the completions on both settings in Fig. 6 and 7. hGCA generates the most accurate geometry compared to all baselines in every scenario according to Table 2. The completions of sparse scans for both the scene and the car show that vanilla GCA and hGCA still generate reasonable completions for a novel object. Because other baselines (SCPNet [20], JS3CNet [21], SG-NN [5]) utilize global features, they create random artifacts when the input is a severely sparse scene (5% and 10% in Fig. 6). Such challenging scenarios are effectively handled with local generations of GCAs, demonstrating superior performance on generalization. As it is impossible to control the input quality or provide accurate object-wise segmentation in actual scans, the robustness of GCA may pave the way toward practical large-scale scene generation.

Varying Number of Input Scans In addition to random samples, we test a more realistic variation of densities, collecting simulated scans exhibiting occlusions. We train models with five and ten scans on synthetic scenes (Karton City and CARLA [6]) and evaluate results on inputs with different numbers of scans. hGCA can also stably create scenes given various numbers of input scans.

We report quantitative results in Table 3 and visualize random samples in Fig. 13 for the extreme case, where only a single scan is provided. Quantitatively, hGCA outperforms other baselines by a large margin in LiDAR Resim and shows competitive (second best) performance on IoU and Street CD, where the best method differs depending on the dataset. While hGCA demonstrates superior generalization performance to a single scan compared to other baselines, it suffers from degradation with the lack of evidence in the input. For example, in the second column of Fig. 13, the input scan provided in a limited height range results in the ambiguity between the facade of the building and the fence.

We also test our methods by completing dense accumulated scans. For CARLA, we accumulated 80 nearby scans from a random pose; for Karton City, we gathered all the scans, which have an average of 132 scans, as input. We report quantitative results in Table 4 and visualize completions randomly in Fig. 16. hGCA outperforms previous methods on all reconstruction methods in Karton City and performs competitively (second best) in CARLA. While SG-NN reports best results on CARLA, we observe that SG-NN struggles to create scenes beyond the sensor range, such as faces of buildings (first column) or trees or cars (second column) in Fig. 16. In contrast, hGCA successfully generates reasonable geometry in our qualitative examples.

[1] Clickable link to asset on sketchfab.com

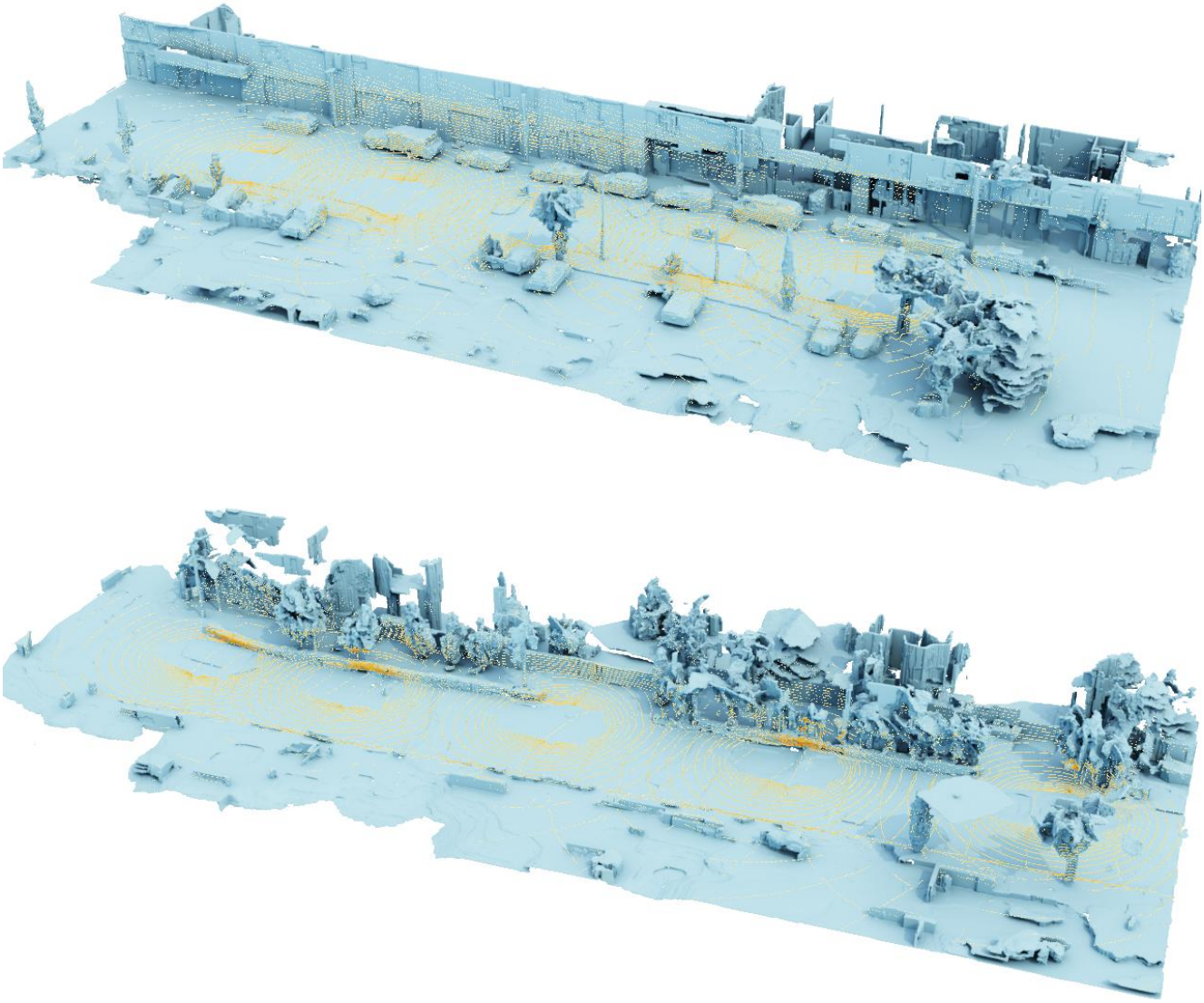


Figure 5. Visualizations on real-world nuScenes dataset on 100m scenes. Yellow spheres indicate input. hGCA is spatially scalable, completing this whole scene (100 meters) at high resolution on a single 24GB GPU without additional tricks.

		CARLA				Karton City						
Method	Representation	High LiDAR ReSim			IoU	High LiDAR ReSim			IoU	Street CD		
		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑
ConvOcc	implicit	21.73		-	12.05	16.6		-	22.05	25.20		-
SCPNet	20cm	9.46		-	38.94	8.23		-	55.01	7.37		-
JS3CNet	20cm	8.70		-	**43.04**	7.20		-	57.05	7.65		-
	10cm	8.49		-	32.78	6.33		-	62.21	**5.48**		-
SG-NN	10cm	9.53		-	37.55	7.11		-	60.62	6.44		-
GCA	20cm	8.37	8.82	1.89	39.07	5.60	5.86	1.40	**63.69**	6.17	7.47	3.21
	10cm	9.16	10.23	4.14	34.73	7.37	7.77	2.14	57.33	8.00	9.18	3.89
cGCA	implicit	9.27	10.10	**4.45**	31.56	6.63	7.06	**3.01**	47.33	10.09	11.83	**7.99**
hGCA	10cm	7.97	8.25	1.19	40.05	**5.16**	5.29	0.91	63.29	6.27	6.93	1.43
	implicit	**7.94**	8.22	1.39	40.47	5.18	5.30	0.96	59.97	5.99	6.67	1.27
input		12.72		-	19.26	12.87		-	21.77	10.14		-

Table 3. Quantitative results on CARLA and Karton City with a single scan given as input. All results except IoU are multiplied by 10 in meter scale. LiDAR Resim and Street CD evaluates the fidelity of completion and TMD measures the diversity of generation. High LiDAR Resim uses high elevation LiDAR to evaluate the extrapolation. IoU is computed with ground truth geometry.

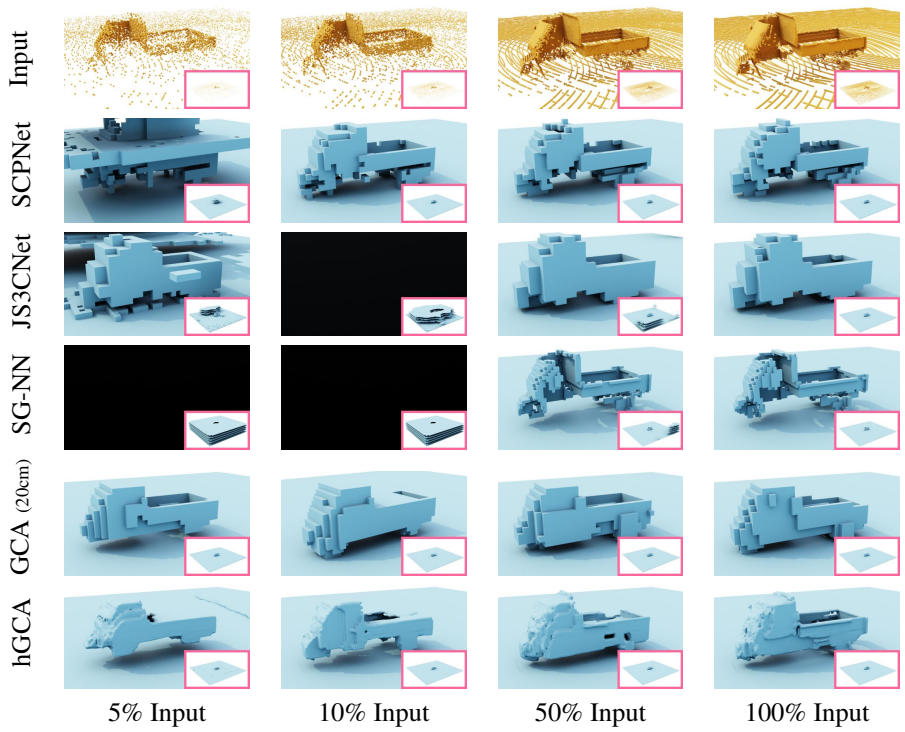


Figure 6. Ablation study on a novel three-wheeler completion by varying density of 5 scans. Inset shows wide range view of the completion. Locality of GCA's enable generalization to sparse input producing stable completions, while method that only utilize global features fail.

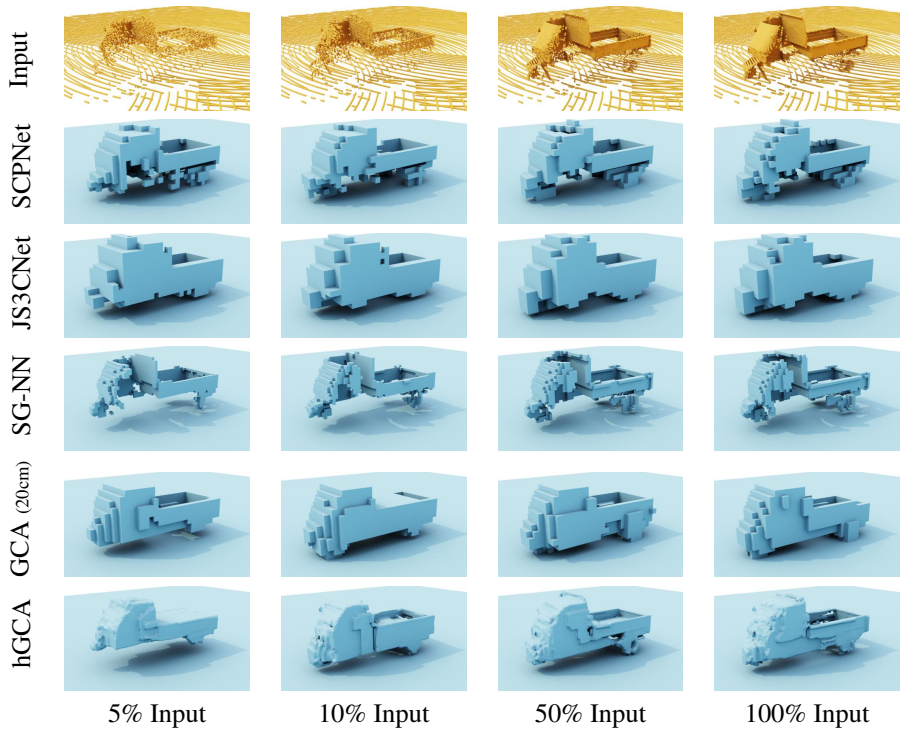
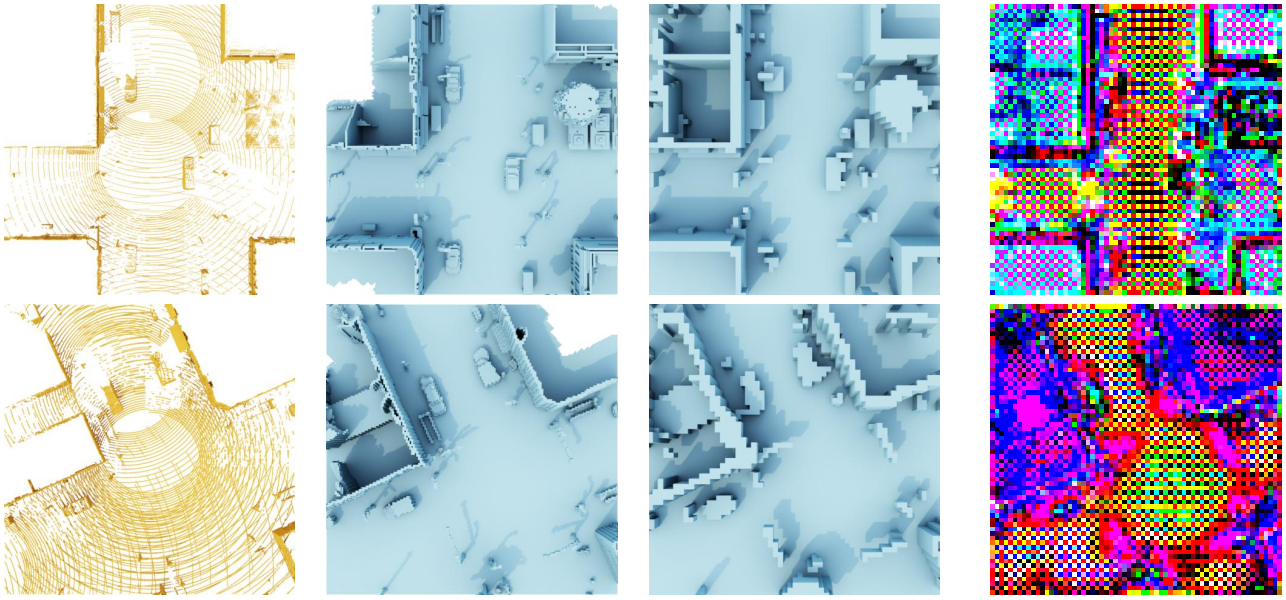


Figure 7. Ablation study on a novel three-wheeler completion by varying density of 5 scans only for the car asset.

Method	Representation	CARLA				Karton City						
		High LiDAR ReSim			IoU	High LiDAR ReSim			IoU	Street CD		
		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑
ConvOcc	implicit	13.4		-	17.81	8.35		-	27.36	13.4		-
SCPNet	20cm	6.03		-	52.10	4.18		-	75.61	3.09		-
JS3CNet	20cm	6.41		-	54.32	5.37		-	65.32	3.59		-
	10cm	5.49		-	48.15	3.45		-	75.44	1.93		-
SG-NN	10cm	**4.20**		-	**59.17**	3.18		-	76.76	1.84		-
GCA	20cm	5.31	5.54	1.45	56.18	3.79	3.83	0.40	79.76	2.64	2.91	0.76
	10cm	5.35	5.85	1.94	50.97	3.17	3.24	0.64	77.47	2.02	2.28	0.84
cGCA	implicit	6.43	6.82	**2.34**	40.72	3.92	3.97	**0.67**	66.86	3.00	3.28	**1.36**
hGCA	10cm	4.54	4.64	0.84	58.78	2.95	2.98	0.35	**81.87**	1.73	1.84	0.47
	implicit	4.36	4.46	0.89	56.85	**2.89**	2.92	0.39	75.48	**1.54**	1.66	0.38
input		5.68		-	45.77	5.09		-	62.36	5.58		-

Table 4. Quantitative results on CARLA and Karton City with a many scans (CARLA: 80 scans, Karton City: average of 132 scans) given as input. All results except IoU are multiplied by 10 in meter scale. LiDAR Resim and Street CD evaluates the fidelity of completion and TMD measures the diversity of generation. High LiDAR Resim uses high elevation LiDAR to evaluate the extrapolation. IoU is computed with ground truth geometry.



(a) Input (b) Completion (c) Rough dense occupancy (d) PCA visualizations of BEV feature

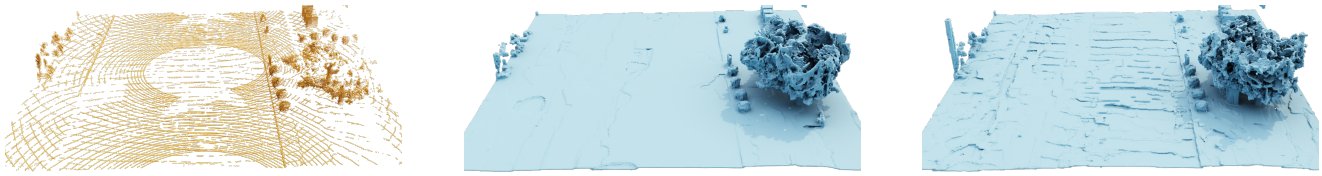
Figure 8. Planner with $z_r = 4$ visualization. From left to right: 5 scan input from Karton City, completion (20cm^3 resolution), rough dense occupancy O_r from planner, BEV feature f_{BEV} visualization using PCA.

A.3. Planner Feature Visualization

For further understanding of the planner, we provide visualizations of outputs and features of planner. Fig. 8 shows the input, completion of GCA equipped with planner, rough dense occupancy O_r of planner, and PCA visualizations of BEV feature. For the PCA visualization, we project the 2D BEV features of the planner to RGB using the first 3 principal axes of PCA. We observe that the final completion follows the rough dense occupancy prediction of the planner. This demonstrates that the planner acts as a memory that persists through the Markov process of GCA and *plans ahead* persisting BEV feature solely with initial state s^0. Also, we observe that BEV feature learns some global context that distinguishes some semantic classes trained without any semantic supervision. We presume that checkerboard artifacts arise due to deconvolution layers of unet [11].

A.4. Effects of LiDAR noise

We investigate the effects of input LiDAR noise on training data. During training on synthetic data, we add Gaussian noise of standard deviation 0.01 in meter scale to the coordinates of each points and add noise to the pitch angle of the pose with standard deviation 0.02 in degree scale to simulate



(a) Input (b) Trained with noise (c) Trained without noise

Figure 9. Completion visualization on Waymo-Open dataset with and without noise. From left to right: input, completion trained with noise, completion trained without noise. Adding noise during training with synthetic data produces cleaner completions on real-world data.

		CARLA				Karton City						
Method	Representation	High LiDAR ReSim			IoU	High LiDAR ReSim			IoU	Street CD		
		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑
ConvOcc	implicit	14.45		-	14.66	9.97		-	25.32	16.91		-
SCPNet	20cm	6.28		-	54.20	4.75		-	70.83	3.29		-
JS3CNet	20cm	6.30		-	52.76	5.16		-	64.93	3.30		-
	10cm	4.87		-	54.80	3.88		-	71.16	2.48		-
SG-NN	10cm	4.76		-	54.99	3.84		-	72.82	2.38		-
GCA	20cm	5.63	5.87	1.24	55.63	3.95	4.02	0.44	77.44	2.79	3.09	0.74
	10cm	6.20	6.71	**2.77**	45.00	3.53	3.69	1.03	71.61	2.41	3.27	2.02
cGCA	implicit	6.50	6.96	2.62	33.21	4.28	4.46	**1.33**	60.42	2.7	3.86	**2.35**
hGCA	10cm	**4.62**	4.75	0.70	**56.41**	**3.17**	3.21	0.45	**77.85**	1.90	2.02	0.53
	implicit	4.71	4.83	0.79	55.20	3.18	3.22	0.51	72.70	**1.65**	1.77	0.41
input		6.14		-	36.63	6.76		-	39.51	5.41		-

Table 5. Quantitative results on CARLA and Karton City with 5 scans given as input, both trained and evaluated without noise. All results except IoU are multiplied by 10 in meter scale. LiDAR Resim and Street CD evaluates the fidelity of completion and TMD measures the diversity of generation. High LiDAR Resim uses high elevation LiDAR to evaluate the extrapolation. IoU is computed with ground truth geometry.

the LiDAR noise produced in data acquisition in real-world. Fig. 9 visualizes the completion of hGCA with and without adding noise during training. We observe that adding noise is crucial in terms of fine sim-to-real generalization, especially on the ground. While we simulated the LiDAR noise with simple ray-casting and Gaussian noises, one could further reduce sim-to-real generalization by employing more sophisticated noise, such as [9].

For completeness, we report quantitative results compared to existing methods without adding noise during both training and validation in Table 5 and 6. Similar to results with noise, hGCA outperforms baselines on reconstruction metrics, demonstrating the superior extrapolation performance of hGCA.

A.5. Space and Time Complexity

In this section, we further analyze space and time complexity of our model. We first investigate the GPU memory usage for hGCA. In Table. 7, we report the GPU memory requirements by varying the completion size in one Waymo scene, visualized in top of Fig. 1. Given an input of size 40 $\times w$ meters, we perform 3 completions and report the maximum GPU memory usage for the coarse completion and the upsampling module. We find that hGCA can scalably generate fine geometry up to 40 $\times$ 120 meters without any tricks on a single 24GB GPU, demonstrating the superior scalability of hGCA by only employing efficient sparse convolutions and planner. We also observe that only 4.8GB is required for the coarse completion on 120 meter scene, demonstrating the efficacy of the planner module. While our ablation study was conducted on only a single scene, we find that GPU memory usage can vary heavily depending on a scene.

For time complexity, we find that coarse completion of our method takes 3 seconds and upsampling takes about 10 seconds (including IO) to create the final mesh in CARLA with 3090 GPU. Indeed our method is slow, whereas other single inference methods (SCPNet [20]) typically take around 0.1 seconds on A100 GPU to create the completion in 20cm voxel resolution. We expect faster inference using half-precision or more recently developed sparse convolution libraries, such as torchsparse [18], but we leave it to future work.

B. Dataset

B.1. Karton City

Karton City is a synthetic city comprised of 20 blocks, obtained from the Turbosquid marketplace[2] for 3D asset. We split 20 blocks into 12/3/5 train/val/test splits and re-combine

[2]Clickable link to asset on turbosquid.com

Method	Representation	CARLA				Karton City						
		High LiDAR ReSim			IoU	High LiDAR ReSim			IoU	Street CD		
		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑		min. ↓	avg. ↓	TMD ↑
ConvOcc	implicit	13.67		-	15.11	9.00		-	26.12	15.24		-
SCPNet	20cm	5.90		-	56.88	4.29		-	74.84	2.83		-
JS3CNet	20cm	6.30		-	52.76	4.99		-	66.63	3.02		-
	10cm	4.41		-	58.38	3.40		-	74.32	1.99		-
SG-NN	10cm	4.31		-	58.05	3.20		-	76.52	1.84		-
GCA	20cm	5.46	5.65	1.09	58.12	3.81	3.86	0.31	81.02	2.62	2.88	0.62
	10cm	5.68	6.21	2.45	48.27	3.08	3.19	0.69	76.54	1.94	2.49	1.38
cGCA	implicit	6.26	6.75	2.45	35.16	4.00	4.13	**1.00**	63.79	1.87	2.62	**1.46**
hGCA	10cm	**4.28**	4.39	0.67	**59.00**	**2.96**	2.99	0.34	**81.34**	1.62	1.69	0.44
	implicit	4.36	4.47	0.75	57.34	**2.96**	2.99	0.40	75.34	**1.38**	1.46	0.33
input		5.19		-	44.02	5.33		-	49.89	4.56		-

Table 6. Quantitative results on CARLA and Karton City with 10 scans given as input, both trained and evaluated without noise. All results except IoU are multiplied by 10 in meter scale. LiDAR Resim and Street CD evaluates the fidelity of completion and TMD measures the diversity of generation. High LiDAR Resim uses high elevation LiDAR to evaluate the extrapolation. IoU is computed with ground truth geometry.

method	40m	60m	80m	100m	120m
Coarse completion	2.8	3.1	3.6	4.4	4.8
Upsampling	5.8	7.0	10.2	12.8	15.7

Table 7. Maximum GPU usage for 40 $\times$ w meter completion on one Waymo scene (top visualization in Fig 1). w denotes the width of the completion in the first row of the table and the unit of GPU memory is GB.

4 blocks in each split randomly to generate 300/30/60 unique train/val/test scenes of size 140 $\times$ 140 meters. Of the 12/3/5 split, 7/2/3 are city blocks and 5/1/2 are suburban blocks. For realistic environment of scenes we re-combine city and suburban blocks separately, and generate 200/20/40, 100/10/20 scenes for train/val/test splits of city and suburban scenes, respectively, which we visualize in Fig. 10. We place ShapeNet [2] cars on each side of the main street to simulate parked cars. The number of cars on each side of the street follow a Poisson distribution with lambda 2, with maximum 7 cars. We uniformly distribute the location of the cars and move them if collisions occur. The cars placed on the street follow the dataset split from [22] and remove meshes that contain unused vertices which makes it hard place the cars in the desired location, thus having total of train/val/test split of 2383/339/670 cars. For each scene, we run three simple trajectories visualized in Fig. 10 and randomly select 4615/30/180 center poses used for train/val/test split, respectively. For each center pose, we create accumulated 5/10 scans for input by accumulating the scans from the center pose and 4/9 other poses, resepectively. For obtaining ground truth implicit function, we sample 4,000,000 points from ground truth mesh with random noise variance 0.03 and 0.1 in meter scale, total of 8,000,000 point-distance pairs.

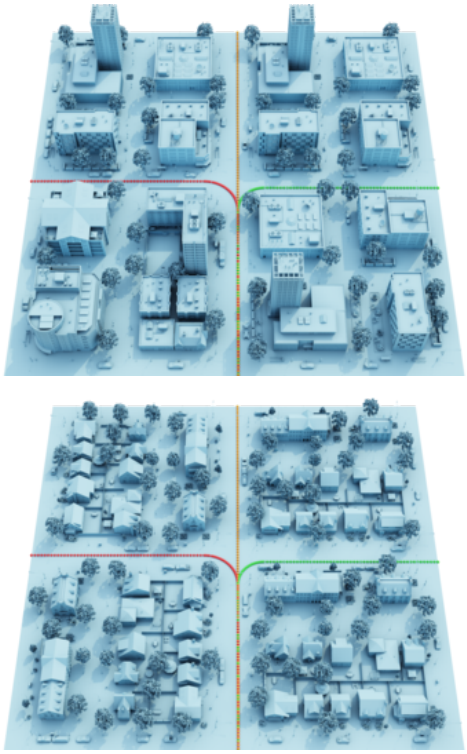


Figure 10. Karton City visualization. City (top) and suburb (bottom) scenes are visualized with three simple trajectories (red, yellow, green) for simulating a drive.

B.2. CARLA

CARLA [6] is an open source driving simulator with diverse environments. We use 5/1/1 towns as train/val/test split with randomly placed static vehicles and run 10 drives for each town to obtain the scans. For each town, we run 10 drives and randomly select 3500/30/180 center poses used

for train/val/test split, respectively. For each center pose, we create accumulated 5/10 scans for input by accumulating the scans from the center pose and 4/9 other nearby poses, respectively. In CARLA, obtaining ground truth mesh is non-trivial. Therefore, we leverage extra LiDAR sensors other than the LiDAR sensor to collect input scans, to obtain the ground truth geometry. We additionally place 5 LiDAR sensors with high elevation angle, having relative offsets of (0, 0, 0), (0, -9.6, 3), (0, 9.6, 3), (0, -19.2, 3), (0, 19.2, 3) meters from the position of LiDAR sensor placed on simulated ego-vehicle to collect input scans, where x-axis and z-axis refer to the front and up direction of the ego-vehicle. We obtain ground truth surface points by accumulating points from scans acquired from additional LiDAR sensor, visualized in top of Fig. 11. For obtaining implicit function, we sample 4,000,000 points from ground truth surface points with random noise variance 0.03 and 0.1 in meter scale, total of 8,000,000 point-distance pairs. The distance for each points are computed with nearest neighbor against the ground truth surface, since ground truth mesh is not available.

B.3. Waymo-Open

Waymo-Open [17] is a real world dataset for autonomous driving containing sequence of LiDAR scans from a ego-vehicle drive. We randomly sampled 202 scenes and selected 3 center poses for each scene to evaluate the quantitative metrics. For each center pose, we create accumulated scans that serve as input by accumulating the scans from the center pose and 4 other poses within 50 meters. We remove dynamic objects from the point clouds using annotated bounding box tracks for both input and accumulated point clouds. Due to noisy dynamic label annotations, some sparse dynamic points lie after bounding box filtering. Thus, we further perform erosion to the accumulated points in voxel resolution of 10cm^3.

B.4. Lidar Simulation

For all the experiments, we simulate synthetic LiDAR using the LiDAR beam angle and rotation-speed parameters from the Waymo-Open dataset [17]. The Waymo LiDAR captures full 360 degrees and results in range image dimension of 64×2650 pixels. For details on the sensor specification see [16]. To simulate LiDAR on Karton City, we generate a curve by interpolating the ego-vehicle poses and simulate a rotating beam along this curve. This simulation, thus correctly captures rolling shutter effects. To obtain ground truth points in CARLA, we use 512 channels LiDAR with field of view (-30°, 30°) and range of 75 meters.

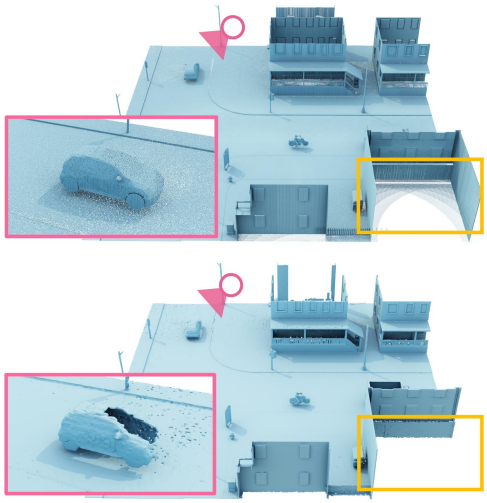


Figure 11. Visualization of CARLA dataset. Top: Ground truth surface points obtained from additional sensors. For visualization of density, we intentionally render with low density. Bottom: Decoded implicit function for supervising upsampling module. Inconsistent, sparse regions (yellow) lead upsampling module to unstable training. Thus, to remove sparse regions for supervision, we train only on regions visible from the street, which may be incomplete (pink), but provide dense and accurate geometry for supervision. However, upsampling is a local operation and upsampling stage of hGCA can be trained with incomplete data.

C. Implementation Details

C.1. hierarchical Generative Cellular Automata

Training Upsampling Module. We train the upsampling module by minimizing the log-likelihood of the data distribution, which we defer to cGCA [24] for further details. We train on synthetic data by combining CARLA and Karton City like other methods. However, as mentioned in Sec. B.2, obtaining ground truth mesh is difficult for CARLA, and while the ground truth points obtained from additional sensors may be dense enough for 10cm^3 voxel resolution, we observe sparse surface points in occluded regions from the street, such as interiors of the building (Fig. 11). Naive training of upsampling module with sparse surface points led to unstable training of local implicit latent feature, where upsampling results varied inconsistently depending on training step. Therefore, for training the upsampling module on CARLA, we supervise with ground truth augmented state x on regions visible from the road, as visualized in bottom of Fig. 11, which tend to be dense. We observed that the upsampling module can generalize to complete scenes even with training on incomplete ground truth, since upsampling is a local operation. During training, we train on combined CARLA and Karton City dataset with rate 15% and 85%, which led to stable training.

Neural Network Architecture. For sparse convolution network of our coarse completion and upsampling module, we employ the same MinkowskiUNet [3, 14] as in GCA [23] and cGCA [24], respectively. We additionally append 3D positional encoding with 128 dimension to the features of the input sparse tensor. For planner module, local point net consists of a fully connected layer that transforms the normalized coordinates to 32-dimension feature followed by 3 fully connected residual block and another fully connected layer with a 32-dimension feature output. The residual block consists of 2 fully connected layers with 32 hidden dimensions. After the local pointnet, we add 2D positional encoding to the features and pass it through a 2D UNet[3] [14] to obtain a 2D feature of dimension 128. Lastly, we employ 5 convolutional blocks, which consists of two convolutions of kernel size 3, for obtaining 4 SPADE [12] features that compute the mean and variance per pillar for denormalization and one rough occupancy prediction.

Other Details. We use MinkowskiEngine [3] for sparse convolutions. For all GCAs we use the infusion scheduler of $\alpha^t = 0.15 + 0.005t$, and obtain the last state with additional maximum likelihood estimation instead of randomly sampling. We use Adam [7] optimizer with constant learning rate 5e-4 and clip gradient with maximum norm of 0.5. For our coarse completion model, we use batch size of 6 and for the upsampling module, we crop a scene into quarters and use batch size of 3. We train the low-resolution GCA attached with planner for 400k steps and upsampling cGCA for 300k steps which takes roughly 5/4 days, respectively, with a single 3090 GPU. Note that the two models can be trained independently.

C.2. Baselines

Generative Celluar Automata [23, 24]. We use the official implementation released from the authors[4]. For fair comparison, we use the same hyperparameters as our model. We use cGCA of voxel size 20cm^3.

Convolutional Occupancy Networks [15] We use the 3D grid resolution of 64 version from the official implementation released from the authors[5]. For obtaining occupancy representation, since we cannot obtain watertight mesh for neither Karton City nor CARLA, we make occupancy for point in the sampled point-distance pairs that have distance to surface below 5cm. We additionally sample 100,000 points in the block range uniformly to create unoccupied points.

SG-NN [5]. We compare with the state-of-the-art indoor scene completion network. We use the official implementation released from the authors[6]. We use the SG-NN to predict the occupancy in 10cm^3 voxel resolution.

[3] https://github.com/milesial/Pytorch-UNet
[4] https://github.com/96lives/gca
[5] https://github.com/autonomousvision/convolutional$_o ccupancy_n etworks$
[6] https://github.com/angeladai/sgnn

JS3CNet [21] and SCPNet [20] We compare with JS3CNet [21] and SCPNet [20], state-of-the-art outdoor semantic scene completion methods. We use the official implementation released from the authors[7][8]. We adapt the method to our setting by changing the semantic class output to binary variable representing occupancy. We observe a class imbalance problem during training, where there are much more empty voxels than the occupied ones. We find that weighing the loss 3:1 for occupied to empty cells performs best for both models. While the original semantic scene completion works uses 20cm^3 voxel resolution, we additionally train on 10cm^3 voxel resolution for JS3CNet. For 10cm^3 model, we modify the output resolution to 10cm^3 to match our voxel resolution by adding an extra upsampling layer. For SCPNet, we omit training on 10cm^3 resolution since it did not fit in a single 24GB GPU.

C.3. Other Implementation Details

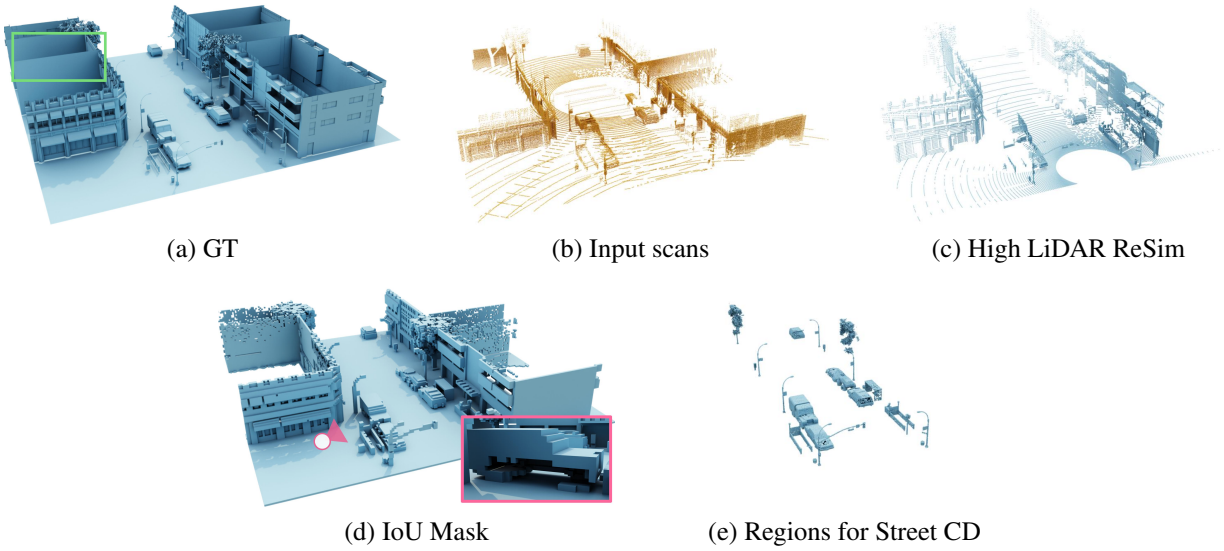
All of our methods are implemented using PyTorch [13]. For all point cloud to voxel conversion, we first round point cloud into 10cm^3 voxels and use floor operation on the coordinates of voxels to create 20cm^3 voxels. For any models except the upsampling stage of hGCA, we train on a dataset combined with CARLA and Karton city having a rate of 50% for each dataset. For methods that utilize unsigned distance fields (cGCA, hGCA), we create mesh in 5cm^3 voxel resolution with marching cubes [8] using the unsigned distance values of the voxels. For IoU and street CD evaluation, we obtain points close to surface by sampling voxels in 5cm^3 resolution that have implicit distance below 0.5. For ConvOcc [15], we evaluate IoU and street CD by sampling from the points created from the mesh, which represents the surface of the completed shape in occupancy representation. We use blender [4] for visualization. For visualizations overlayed with input, such as Fig. 1, we render input points if a point is either in front of a mesh or is less than 0.5 meters back from the mesh in the rendering view.

D. Evaluation Metric

High LiDAR ReSim evaluates the fidelity of the completion beyond the LiDAR range focusing on regions visible from the street, visualized in Fig. 12. It computes the Chamfer distance between a ground truth LiDAR scan and a re-simulated LiDAR scan from a pose distant from the center after the completion. The metric avoided evaluating inconsistent geometry in interior walls of the buildings in ground truth geometry (green in Fig. 12).

Given an origin in the ego-vehicle frame, we set a region of interest R to be a box of 38.4 $\times$ 38.4 meters. We take the scanned input point cloud X within R and generate

[7] https://github.com/yanx27/JS3C-Net
[8] https://github.com/SCPNet/Codes-for-SCPNet



(a) GT (b) Input scans (c) High LiDAR ReSim

(d) IoU Mask (e) Regions for Street CD

Figure 12. Evaluation visualization. (a) Ground truth geometry in Karton City. (b) Input scans. (c) High LiDAR Resim of GT from a novel pose. (d). Ground truth occupied regions for IoU evaluation. (e) Regions for evaluating street CD. High LiDAR ReSim and IoU captures geometry above the input LiDAR range, while it does not capture inconsistent building interiors (green). Street CD is the only metric that can evaluate completion of occluded geometry from road, such as side-walk side of the car.

completion Y. We select two poses $p_1, p_2 \in P$ from the ego vehicle trajectory that enters and leaves the region of interest R. The selected poses are distant from the center, which and inside R, making the re-simulated LiDAR scan far from the input scan taken from the center while making the LiDAR ReSim free of occlusions occurring outside of R. We found only 3 out of 1440 (180 scenes $\times$ 2 poses per scene $\times$ 4 test configurations with Karton City/CARLA dataset with 5/10 input scans) poses overlapped with the input pose, indicating that the selected poses for evaluation are mostly unique. We perform lidar simulation from poses p_t one the completion Y in mesh representation. If the completion is in voxel representation, we convert it into mesh using marching cubes [8] and average the Chamfer Distance between the GT and re-simulated LiDAR scan. For high range LiDAR used from evaluation, we use a 128-beam LiDAR with elevation angle (-30°, 30°), which acquires denser scans than the input captured with 64-beam LiDAR, to cover the range above the input scan, visualized in (c) of Fig 12. Thus the Chamfer distance (CD) for input X is defined by

$$\text{CD}_{ReSim}(X, Y) = \frac{1}{|P|}\Sigma_{p_t \in P}\text{CD}(X_t, Y_t),$$

where X_t is the lidar scan from pose p_t in the region of interest and Y_t is the simulated lidar scan of generation Y from pose p_t. We also evaluate diversity TMD as in [19, 23, 24] for input X is defined as

$$TMD_{ReSim}(X, Y) = \frac{1}{|P|}\Sigma_{p_t \in P} TMD(\{X_t\}, \cup_{1 \le k \le K}\{Y_{t,k}\}),$$

where $Y_{t,k}$ is the k-th completion for input X_t and TMD is defined as the following:

$$\text{TMD}(S_p, S_c) = \frac{1}{|S_p|}\sum_{P \in S_c}\frac{2}{k(k-1)}\sum_{1 \le i < k}\sum_{i < j \le k}\text{CD}(C_i^P, C_j^P),$$

where $P \in S_p$ denotes the partial input and $S_c = C_{1:k}^P$ is the set of completions c_i^P for partial input P.

D.1. IoU

IoU is evaluated on the visible regions in 20cm^3 voxel resolution from the street following the previous semantic scene completion (SSC) works [1, 20, 21]. In contrast to SSC that computes IoU against accumulated LiDAR scans, we compute IoU against ground truth geometry from the visible regions are obtained using high elevation LiDAR, used for High LiDAR Resim, and covers regions beyond the input LiDAR range, visualized in (d) of Fig. 12. To obtain the visibility mask, we perform TSDF fusion [10] from all the poses in a single drive for each block of interest and obtain the TSDF values for the block with grid of voxel resolution 10cm^3. We set the grid to visible only if the TSDF value is bigger than -0.3 and convert the mask into 20cm^3 voxel resolution.

Street CD includes evaluation on geometry completely occluded from the ego-trajectory, such as the sidewalk side of parked cars, visualized in Fig. 12, which neither High LiDAR ReSim nor IoU (pink) can evaluate. On Karton City dataset, where the scene is a simple crossroad junction, we compute Chamfer distance between the generated geometry against GT, only on the objects on the main street. To evaluate objects above the ground, we remove it for both the completion and ground truth by simply thresholding the z-axis with 20cm.

E. Additional Visualizations on Synthetic Scenes

We provide additional visualizations on synthetic scenes in Fig. 13, 14, 15, 16.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019. 1, 14

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 11

[3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 13

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 13

[5] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *CVPR*, 2020. 1, 6, 13

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 6, 11

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 13

[8] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 13, 14

[9] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *CVPR*, pages 11167–11176, 2020. 10

[10] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 14

[11] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 9

[12] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 13

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 13

[14] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). 13

[15] Lars Mescheder Marc Pollefeys Andreas Geiger Songyou Peng, Michael Niemeyer. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020. 13

[16] 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, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019. 12

[17] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2446–2454, 2020. 1, 12

[18] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. Torchsparse: Efficient point cloud inference engine. In *Conference on Machine Learning and Systems (MLSys)*, 2022. 10

[19] Rundi Wu, Xuelin Chen, Yixin Zhuang, and Baoquan Chen. Multimodal shape completion via conditional generative adversarial networks. In *ECCV*, 2020. 14

[20] Zhaoyang Xia, Youquan Liu, Xin Li, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, and Yu Qiao. Scpnet: Semantic scene completion on point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 1, 6, 10, 13, 14

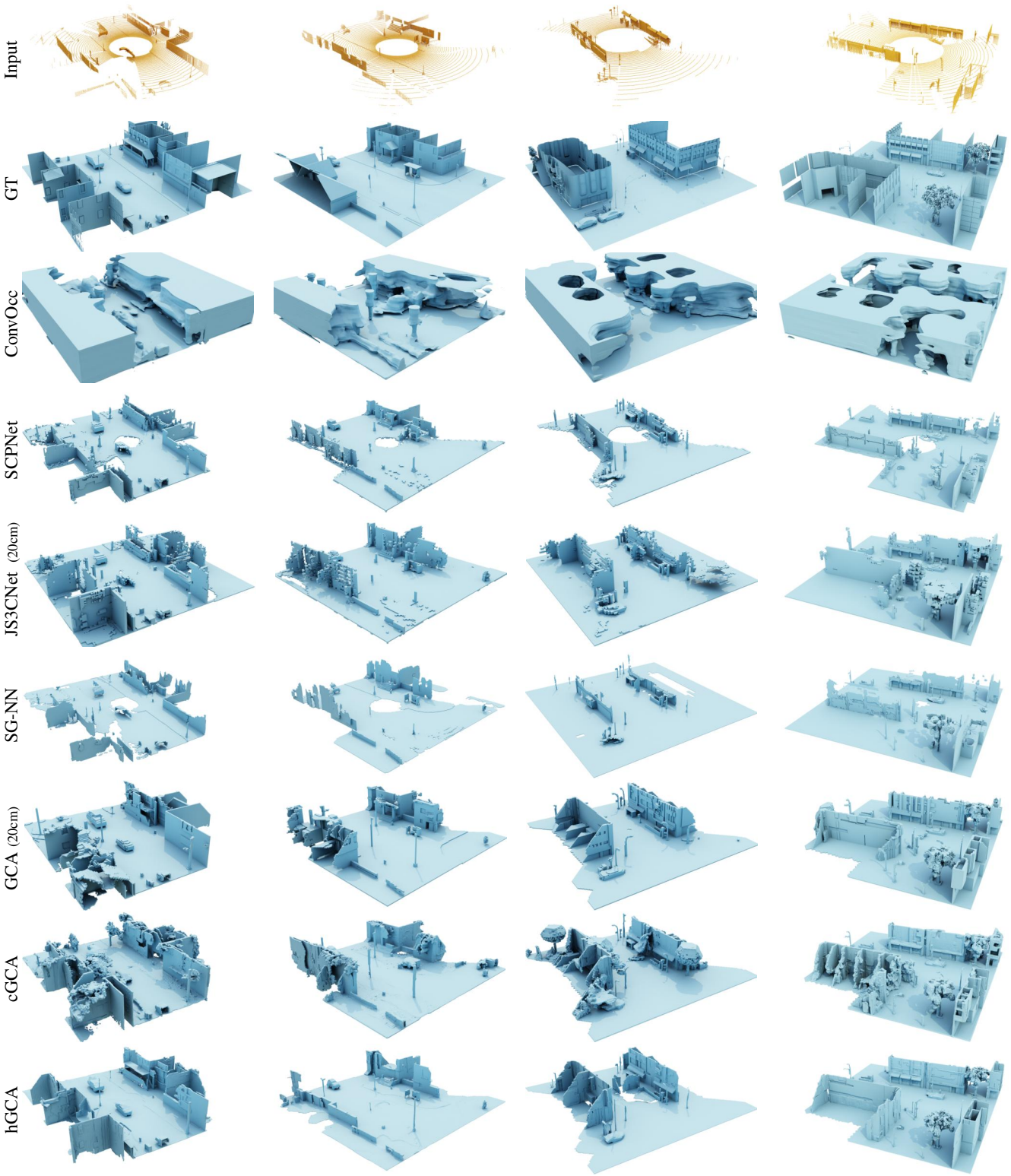


Figure 13. Visualizations on CARLA (first 2 columns) and Karton City (last 2 columns) from a single scan. Scenes were randomly chosen.

[21] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *AAAI*, pages 3101–3109, 2021. 1, 6, 13,

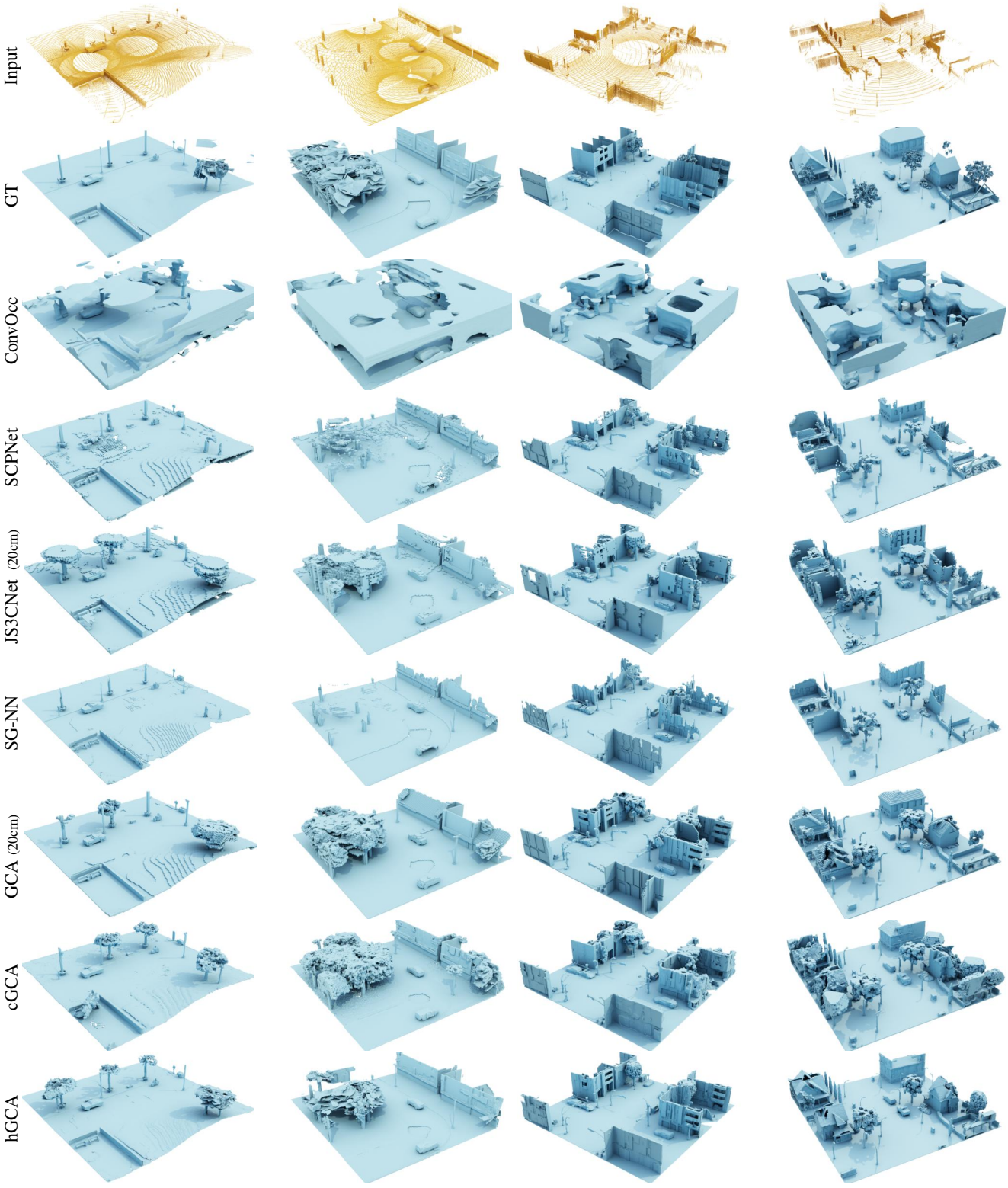


Figure 14. Visualizations on CARLA (first 2 columns) and Karton City (last 2 columns) from 5 scans. Scenes were randomly chosen.

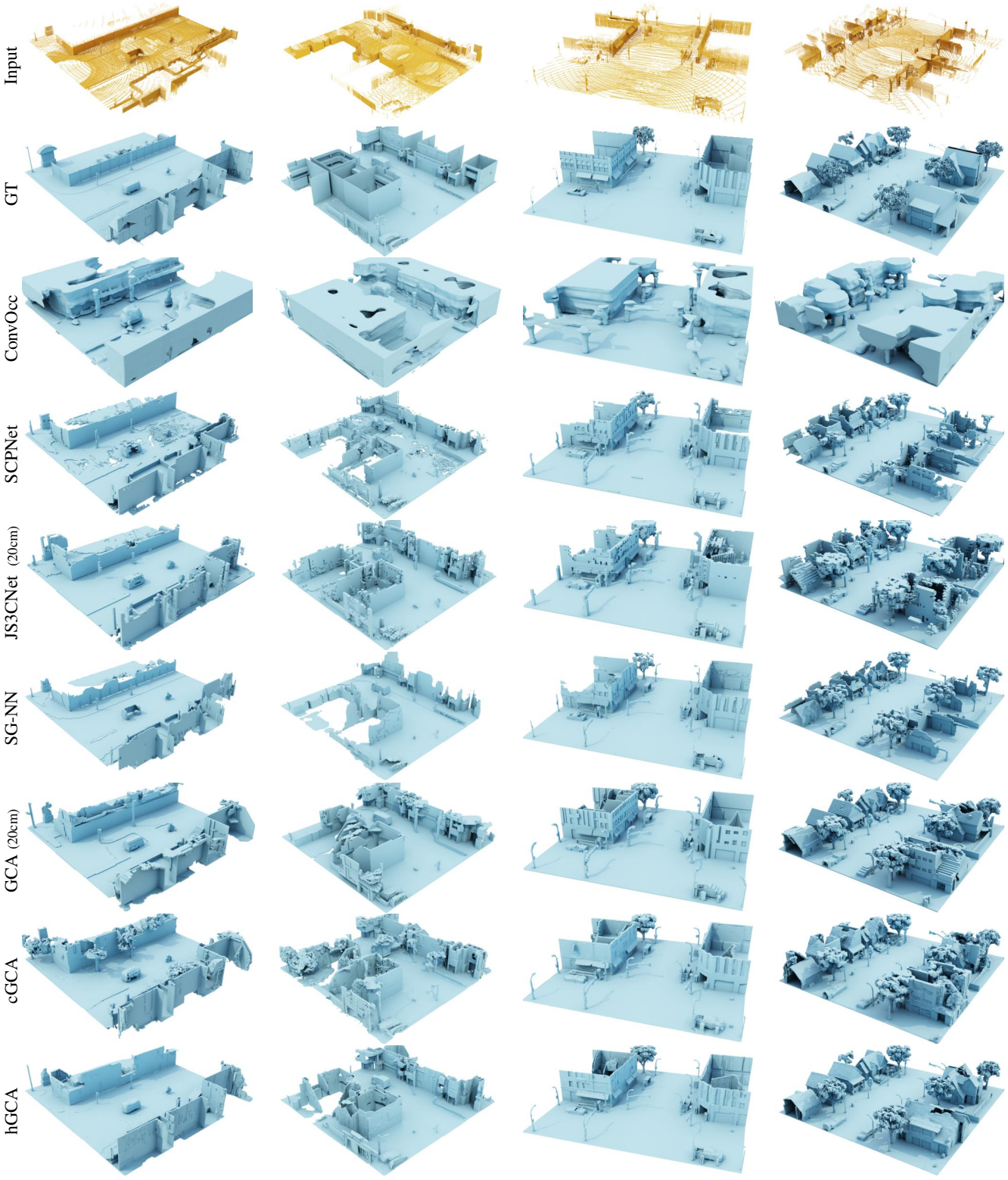


Figure 15. Visualizations on CARLA (first 2 columns) and Karton City (last 2 columns) from 10 scans. Scenes were randomly chosen.

[22] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. *arXiv*, 2019.

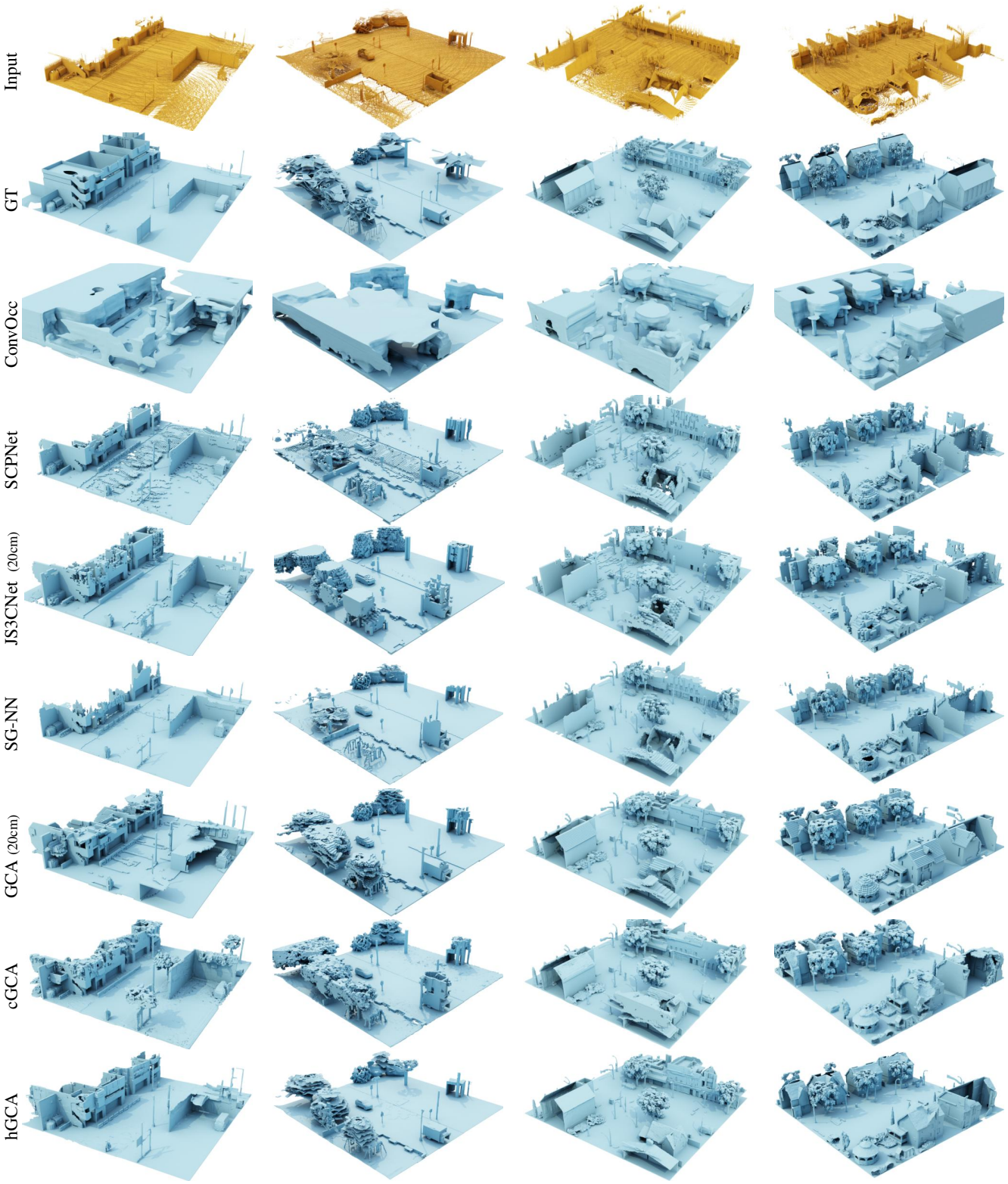


Figure 16. Visualizations on CARLA (first 2 columns) and Karton City (last 2 columns) from many accumulated scans. Scenes were randomly chosen.

11

[23] Dongsu Zhang, Changwoon Choi, Jeonghwan Kim, and Young Min Kim. Learning to generate 3d shapes with generative cellular automata. *arXiv preprint arXiv:2103.04130*, 2021. 13, 14

[24] Dongsu Zhang, Changwoon Choi, Inbum Park, and Young Min Kim. Probabilistic implicit scene completion. *ICLR*, 2022. 12, 13, 14