Appendix

6. Licenses

Below in Table 5 the licenses of the code and assets we make use of are listed. Neo360 is listed because we use its re-implementation of PixelNeRF.

Table 5. **Licenses.**

Item	License
CARLA code	MIT
CARLA assets	CC-BY
NeRFStudio	Apache-2.0
PixelNeRF	BSD-2-Clause
SplatterImage	BSD-3-Clause
6Img-to-3D	BSD-3-Clause
Neo360	Non-commercial attribution

7. Dataset Details

7.1. Extended Dataset Description

The static (3D) dataset encompasses 212k inward—and outward-facing vehicle images, while our dynamic (4D) dataset contains 16.8M images from 10k trajectories, each sampled at 100 points in time with egocentric and exocentric images. Data for the static dataset is collected from 2002 scenes, and for the dynamic dataset, from 498 scenes. Because the static and the dynamic datasets differ in amount of vehicles that are equipped with sensors they differ in their composition as highlighted in Table 6.

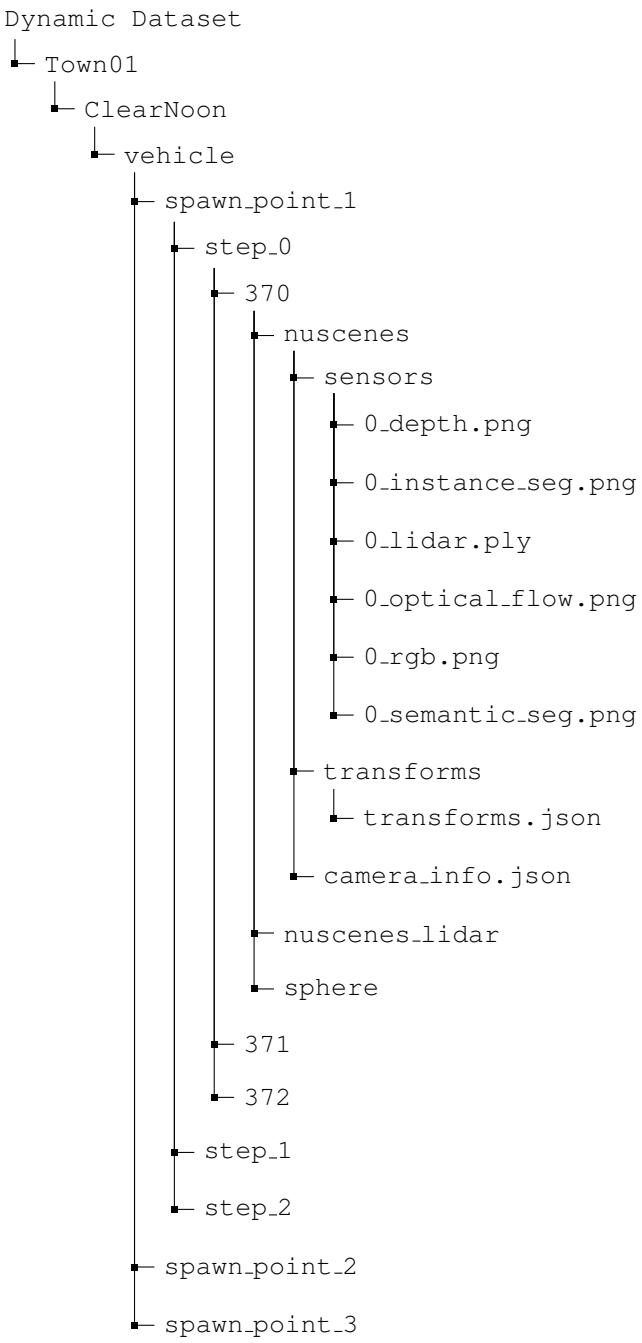
Table 6. **Number of Views.**

Dataset	Egocentric	Exocentric
Static (all)	12k	200k
Static (per vehicle)	12k	200k
Dynamic (all)	6.3M	10.5M
Dynamic (per vehicle)	300k	500k

The uncompressed static dataset has a total size of 437 GB and took 132 hours of GPU time to be generated. Per scene, this corresponds to a size of 0.218 GB and a generation time of 4 minutes. The uncompressed dynamic dataset is 1673 GB large and took 390 hours of GPU time to be generated. Since the dataset contains images from 498 scenes and 21 vehicles per scene this results in 10458 sequences. Each sequence with a length of 100 timesteps has a size of 0.160 GB and required 2.23 minutes to generate.

7.2. Directory Setup

Each of the datasets (static and dynamic) is organized in the following way: towns, weather, ego vehicle type, ego-position (spawn point), timesteps, vehicles in the scene, and finally folders containing the actual sensor measurements, transforms, and camera information.



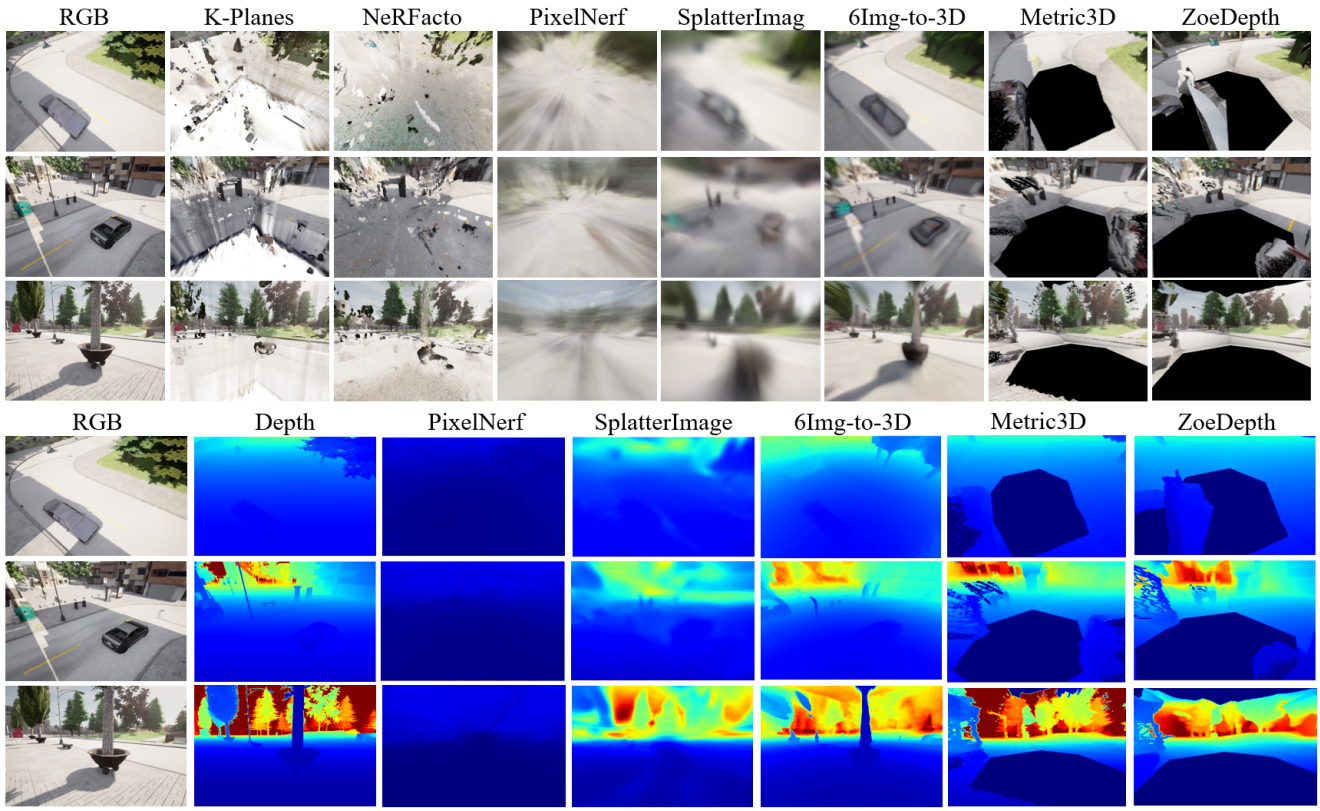


Figure 6. Qualitative Results for the single-shot few image scene reconstruction methods.

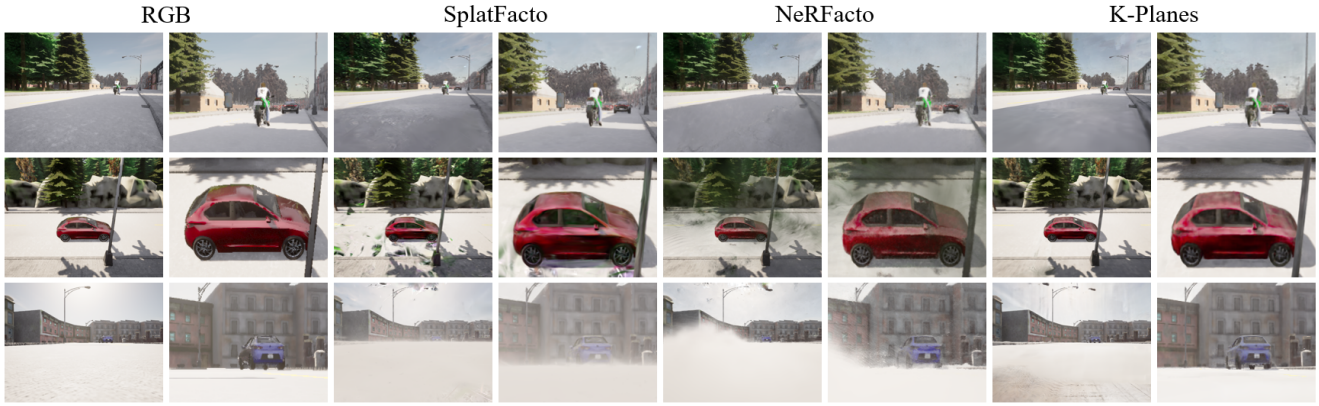


Figure 7. Qualitative Results for the multi-view per scene optimization methods.

8. Benchmark Details

8.1. Qualitative Results

Multi-view Novel View Synthesis. Figure 7 compares the qualitative results of Splatfacto, Nerfacto, and K-Planes. Our analysis shows that K-planes generalizes best both quantitatively and qualitatively, as demonstrated by the minimal presence of floaters. Interestingly, SplatFacto significantly outperforms both other methods on the training set but performs worst on the test set, as shown in Table 7. We hypothesize that K-Planes' planar representation provides geometric regularization that enhances generalization performance.

Single-shot Few-Image Scene Reconstruction. In Figure 6, the methods performing single-shot few-image scene reconstruction. K-Planes, NeRFacto, and PixelNerf visibly struggle to reconstruct the scene. Where the unpro-

Table 7. **Training and Testing result comparison of Multi-view Novel View Synthesis Methods.**

Methods	Train			Test		
	PSNR↑	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**
SplatFacto [48]	44.019	0.984	0.014	24.458	0.806	0.210
NeRFacto [97]	36.206	0.930	0.091	24.936	0.804	0.227
K-Planes [29]	29.827	0.820	0.254	25.744	0.816	0.239

jected depth maps obtained via Metric3D and ZoeDepth result in pixel values good results are obtained. The few-image SplatterImage and 6Img-to-3D perform reasonably well. Due to their low performance, we do not visualize SplatFacto K-Planes and NeRFacto at the bottom part.

8.2. Training Details

K-Planes We train each of the models for 30k steps on a single Tesla T4 GPU with 16GB of VRAM. We follow the model's default NeRFStudio [97] settings for training. Near and far bounds of the scene are adjusted to 0.1 to 60 to best accommodate the scenes. Additionally, scene contraction is applied. The training took around 1.5 hours per model.

NeRFacto We train each of the models for 30k steps on a single Tesla T4 GPU with 16GB of VRAM. We follow the model's default NeRFStudio [97] settings for training. We disable the model's use of an appearance embedding since those lead to problems during the evaluation, and we also deactivate the camera pose optimization because we already provide the model with ground truth poses. The near and far bounds are set to 0.1 and 60. Each model is trained for a total of 1 hour.

SplatFacto We train each of the models for 30k steps on a single Tesla T4 GPU with 16GB of VRAM. We again follow the model's default NeRFStudio [97] settings for training. The model took a total of 20 minutes to train.

PixelNeRF We train PixelNeRF for 100k steps on a Nvidia A40 GPU with 42GB of VRAM, with an Adam optimizer [51] and a learning rate of 1e-3. Total training time accumulates to five days.

SplatterImage We train SplatterImage for a total of five days across five 3090 GPUs with 24GB of VRAM. During training the supervision images are scaled to 128×128 pixels. We use the multi-input image variant of the model to accommodate all six input views.

6Img-to-3D We train 6-Img-to-3D, following their [34] process, with a Nvidia A40 GPU with 42GB of VRAM for 100 epochs with an Adam optimizer [51], a learning rate of 5e-5, and a cosine scheduler [65] with 1000 warmup steps. Each epoch consists of 1900 steps, each comprising a new scene and three randomly sampled views as supervision, scaled to 64×48 pixels. The total training of the model is five days.

ZoeDepth and **Metric3D** were not fine-tuned using our data. For the single-shot few image reconstruction task, we tested both monocular depth estimation as a baseline. We obtained a depth map for each of the six ego input images resized to 842×842 to fit the model. Since camera intrinsics and extrinsic are known, we can use the depth maps to project the image pixels into space to obtain a colored point cloud (sometimes also referred to as 2.5D). The obtained colored point cloud can now be used to rasterize novel exo views.

9. Leaderboard

We will actively maintain a leaderboard on the project page accompanying our SEED4D paper. We welcome contributions to one of the proposed benchmarks or other submissions using the datasets. Submissions can be made by contacting the first author.

10. Hosting, licensing, and maintenance plan

Hosting. To find the latest hosting information of our datasets please see our project page here.

Licensing. Below in Table 8 the licenses of the code and assets we are publishing are listed.

Table 8. **Own Licenses.**

Item	License
Data generator code	BSD-3-Clause
Static dataset	CC BY-SA 4.0
Dynamic dataset	CC BY-SA 4.0
ArXiv paper	CC BY 4.0

Responsibility Statement We believe that our datasets comply with existing licenses and have adhered to their terms and conditions. Despite our careful attention to these requirements, we acknowledge that any responsibility for any potential rights violations remains solely ours. We take accountability for ensuring that all content and actions are following legal and ethical standards.

11. Data Generation Details

11.1. Carla Towns

The towns available within Carla vary in scenery, road structure, and size, with key characteristics highlighted below:

Town 1: Town 1 is a compact environment divided by a river with several small bridges. The road network includes numerous T-junctions and a variety of buildings, both residential and commercial, surrounded by coniferous trees.

Town 2: Town 2 consists of a mix of residential and commercial areas, including a central park, apartment buildings, a church, and a gas station. The road network is composed of T-junctions and tree-lined streets.

Town 3: Town 3 is an urban area featuring a central roundabout, raised metro tracks, and a diverse mix of commercial and residential buildings. The road network includes four-way junctions, T-junctions, an underpass, overpasses, and cul-de-sacs.

Town 4: Town 4 is a small town with a ring road in a "figure of 8" configuration that includes an underpass and overpass. The town features commercial and residential buildings, tree-lined streets, nearby snow-capped mountains, and a pedestrian shopping arcade.

Town 5: Town 5 is an urban setting with multilane roads and a raised highway forming a ring road. The layout includes commercial buildings, a construction site, and a large carpark, with roads passing beneath one of the buildings.

Town 6: Town 6 is a low-density area with wide 4-6 lane roads interconnected by slip roads and junctions, including Michigan Left configurations. The layout features designated turning lanes and cul-de-sacs.

Town 7: Town 7 represents a rural area with cornfields, barns, grain silos, and windmills. Its road network is simple, with unmarked roads, a small residential street, and a short bridge over a water body.

Town 10: Town 10 is an urban grid layout with a mix of junction types, including yellow-box intersections and dedicated turning lanes. The town features waterfront promenades, tree-lined boulevards, skyscrapers, and public buildings such as a museum.

More information about the Carla simulator can be found in the official Carla documentation [25].

11.2. Camera Poses

The algorithm to obtain the spherical Fibonacci lattice is described in detail in Algorithm 1. The procedure equally spaces points on a half-disk. The obtained points are then translated into Carla world coordinates. To obtain the proper camera orientations, we introduce the procedure presented in Algorithm 2.

Algorithm 1 Exocentric camera coordinates

procedure CREATE_SPHERE(N) ▷ N points
  $\phi = 3\pi - \sqrt{5}$
  $ys \leftarrow$ linspace(0, 1, N)
  $points \leftarrow$ empty list
  $idx \leftarrow 0$
  for `y in ys` **do**
    $x = \cos(\phi \cdot idx) \cdot \sqrt{1 - y^2}$
    $y = \sin(\phi \cdot idx) \cdot \sqrt{1 - y^2}$
    $z =$ y
    $points[idx] = [x, y, z]$
    $idx = idx + 1$
  end for
  return $points$ ▷ dim: N x 3
end procedure

Algorithm 2 Exocentric camera orientation

procedure CREATE_SPHERE($points$)
  $pitchs, yaws \leftarrow$ empty lists
  $idx \leftarrow 0$
  for `point in points` **do**
    $x, y, z \leftarrow point$
    $pitch = \arcsin(z)$
    $yaw = sign(x) \cdot \arccos(\frac{y}{x^2+y^2})^{0.5}$
    $pitchs[idx], yaws[idx] = pitch, yaw$
    $idx = idx + 1$
  end for
  return $pitches, yaws$
end procedure

12. Style transfer

We experimented with existing style transfer methods to reduce the domain gap between Carla and NuScenes' images. The results in Figure 8 are obtained using a CylceGan-based framework as proposed in [49]. The checkpoint of our trained model will be made available.

13. Dataset Visualization.

All full RGB images are paired with depth maps, optical flow, segmentation maps, and instance segmentation images. Since all values are ground truth, they can, for example, be used to generate a colored 3D point cloud using the camera's extrinsics and intrinsics, as shown in Figure 9. The sensory setup for an egocentric view is visualized in 10. Figure 11 and Figure 12 the static ego–exo dataset is visualized. Figure 13 and Figure 14 display the dynamic ego–exo dataset.

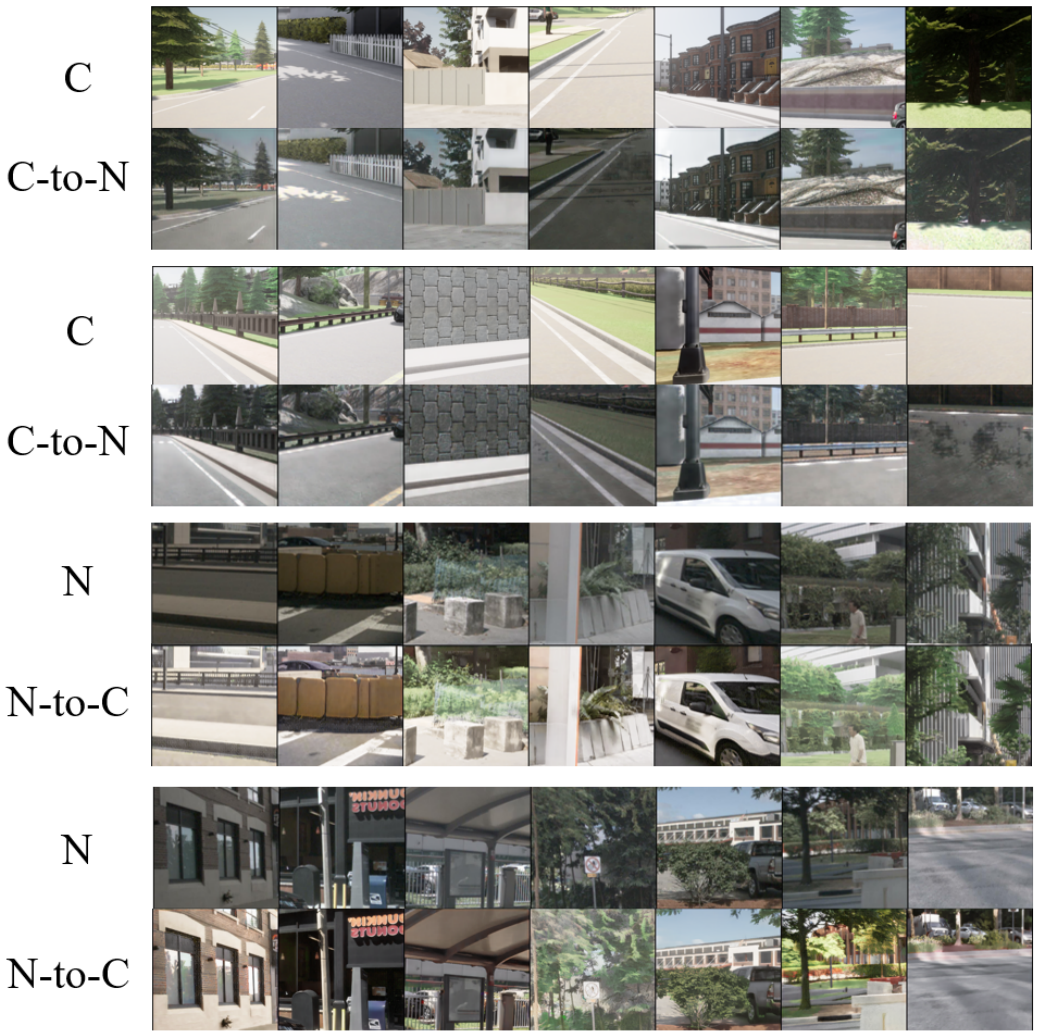


Figure 8. Example style transfer results. 'C' denotes Carla images, 'C-to-N' indicates a transfer from the Carla domain to the NuScenes domain. 'N' indicates NuScenes views, and 'N-to-C' signifies images transferred from the NuScenes domain into the Carla domain. Note: The cycle consistency step, into the original domain, is not illustrated here.

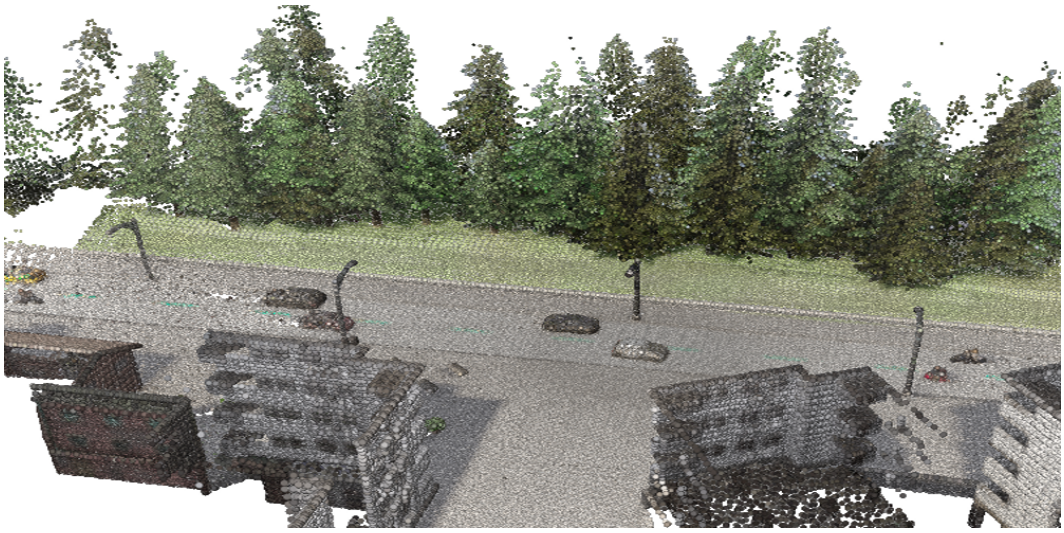


Figure 9. Colored 3D point cloud generated from RGB images, depth maps, camera intrinsics, and extrinsics.

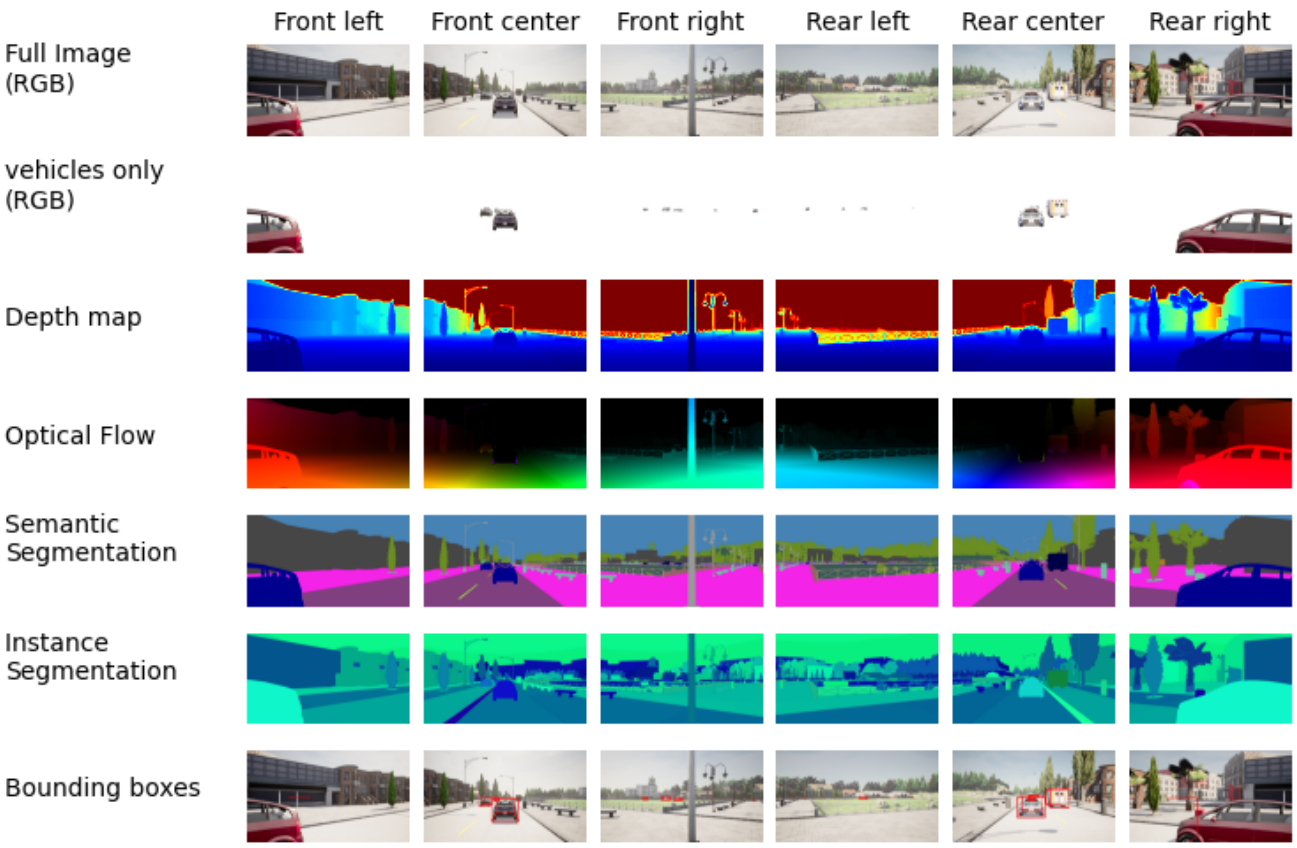


Figure 10. An overview of six egocentric cameras and their associated sensor measurements.

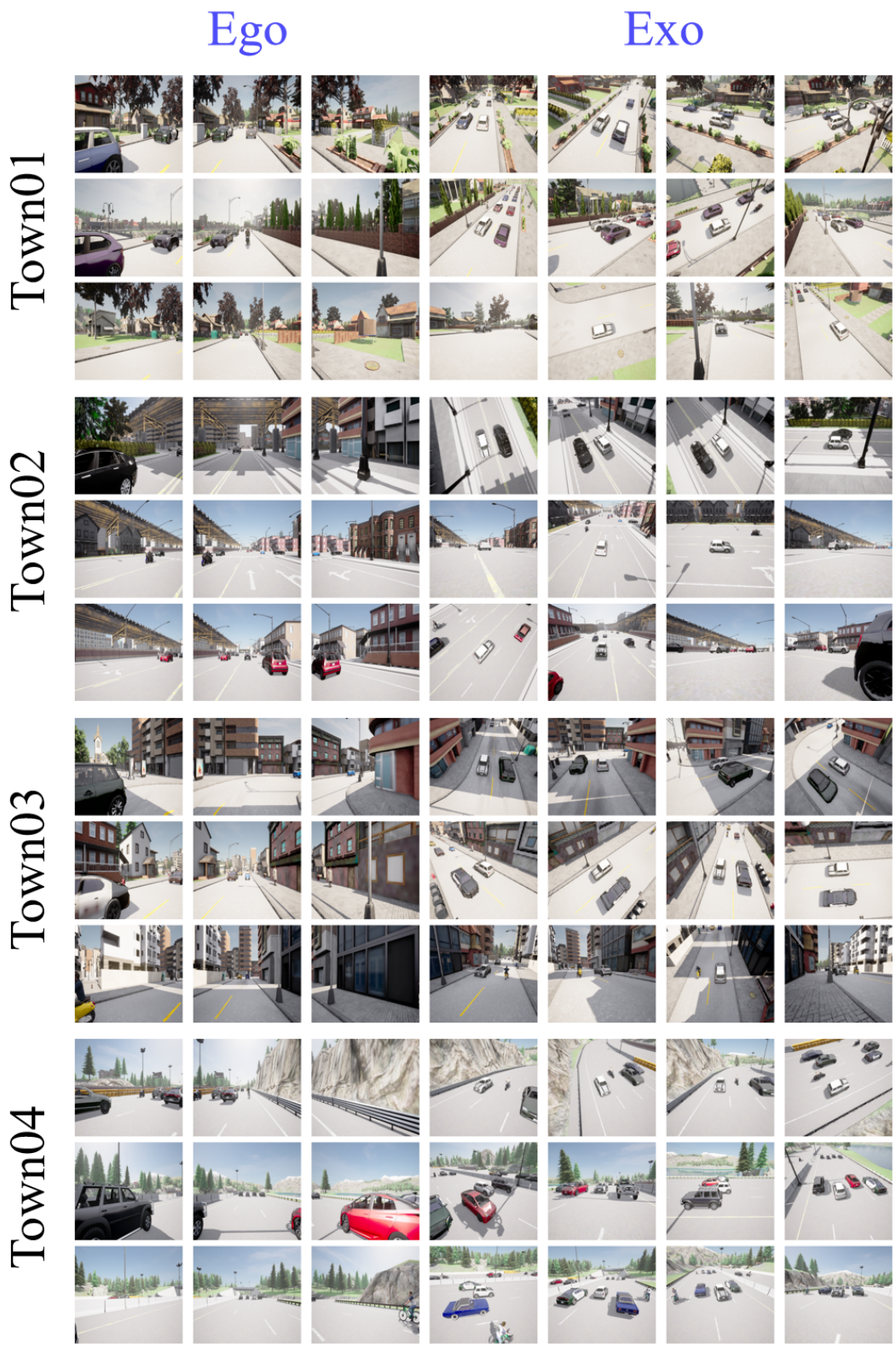


Figure 11. Samples from the Static Ego–Exo Dataset showing towns 1 to 4. The egocentric images show front left, front center, and front right views. The exocentric views are randomly sampled.

Figure 12. Samples from the Static Ego–Exo Dataset showing towns 5 to 7 and 10HD. The egocentric images show front left, front center, and front right views. The exocentric views are randomly sampled.

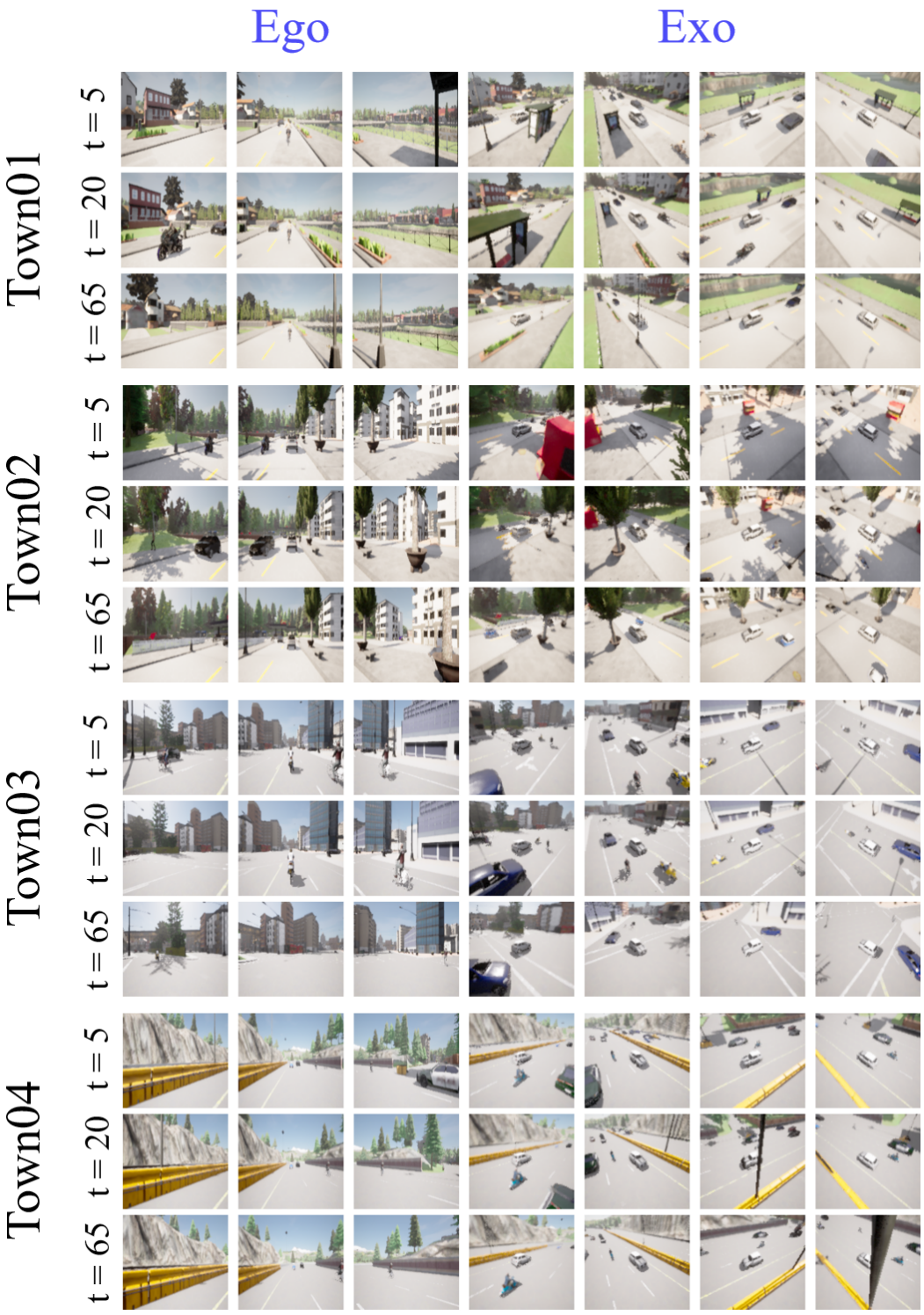


Figure 13. Samples from the Dynamic Ego–Exo Dataset showing towns 1 to 4 for timepoints 5, 20, and 65. The egocentric images show front left, front center, and front right views. The four exocentric views have the same relative pose across all samples.

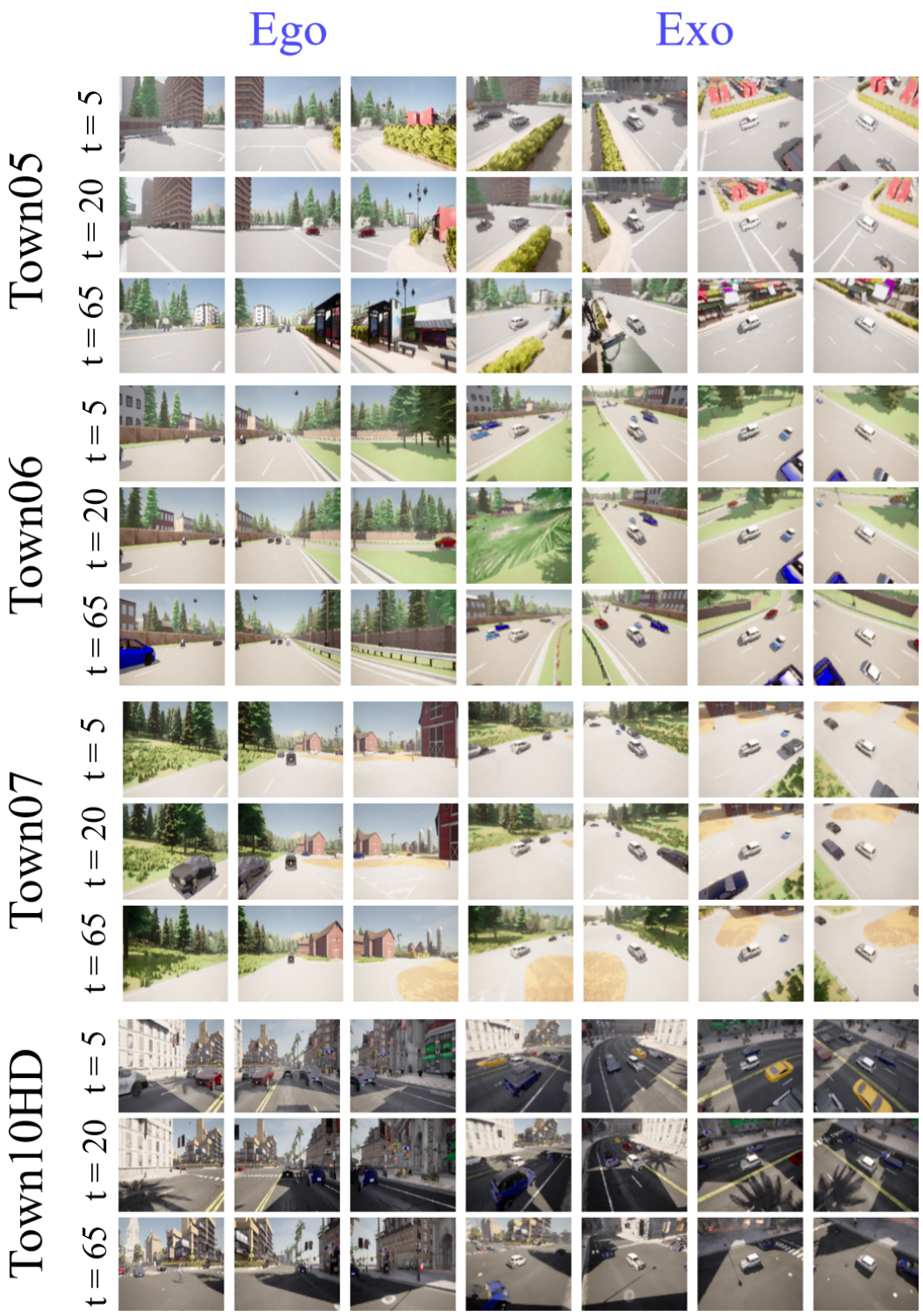


Figure 14. Samples from the Dynamic Ego–Exo Dataset showing towns 1 to 4 for timepoints 5, 20, and 65. The egocentric images show front left, front center, and front right views. The four exocentric views have the same relative pose across all samples.