Supplementary Materials for

A Framework for Generating Dangerous Scenes for Testing Robustness

Anonymous Author(s)

A Datasets

A.1 Visualization

Our dataset is hosted on the Google drive[1], where contains all the DANGER-vKITTI1 gifs generated by the dataset without the raw data for the parameter sets shown in Table S2. You can navigate to `./DANGER-vKITTI1/main.html` to check all gifs for different parameter sets. vKITTI2 only generates parameter sets in Table S1, but both gifs and images are available in the folder.

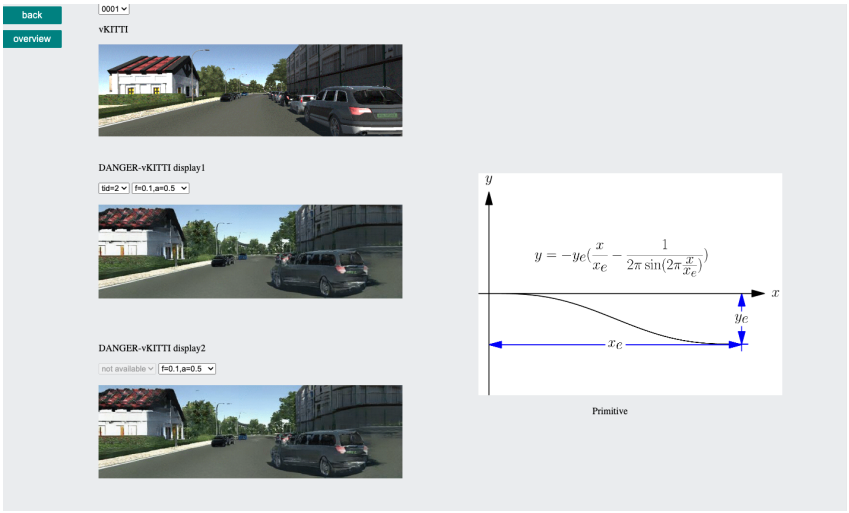


Figure S1: Visualizer for DANGER-vKITTI1

	Exit Parking	Cut-in Opposite	Slalom Lane Change	Cut-in	Braking
DANGER-vKITTI	tid70_epp_xe5_ye-2.2	tid0_cio_xe40_ye15	tid7_lc_f0pt12_A1pt5	tid2_ci_xe10_ye-5	tid16_br_t010_dt4_g0pt35
DANGER-vKITTI2	tid70_epp_xe4_ye-2	tid0_cio_xe37_ye12	tid7_lc_f0pt1_A1pt0	tid1_ci_xe10_ye3	tid16_br_t02_dt2pt5_g0pt5

Table S1

A.2 Datasets Generation Code

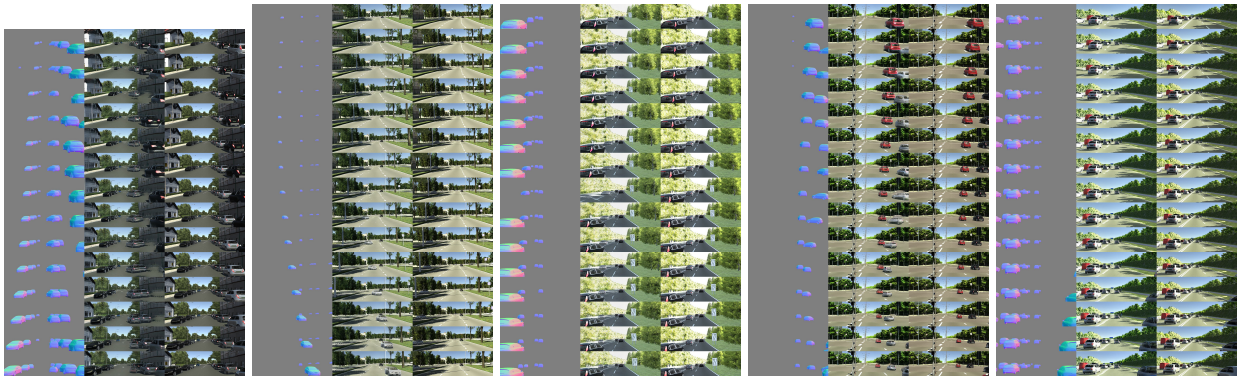
Our Code is open sourced on the Colab server in Jupyter Notebook format. Here we provide the PDF version of the original notebook. The functionality of each file are as follows:

[1] https://drive.google.com/file/d/14ZjBi3ewvf2J92aKImEJgQ2j-ZzhqA_7/view?usp=sharing

- `./DANGER_Design_JSON.pdf` Load metadata from vKITTI dataset, and create modified JSON files for 3D-SDN.
- `./DANGER_CUDA9-3D-SDN.pdf` Hardware setup for 3D-SDN and inference for the test.JSON obtained from the above step.
- `./DANGER_Eval_Score.pdf` Qualitative evaluation for the generated dataset and plot generation.

A.3 Datasets Sample

Posted at the end, see Figure S10.



(a) 0001 **(b)** 0002 **(c)** 0018 **(d)** 0006 **(e)** 0020

Figure S2: Overview of generated datasets.

A.4 Scene Editing Transformation

We only show the data generation and coordinate system conversion pseudocode applied in slalom lane change, see Section B.1 for a detailed visualization.

The following relevant passages from the original text:

> We assume that the original driving trajectory of the target vehicle object is a straight line. Given a image $\mathcal{I} \in R^{W \times H \times 3}$ defined by `scene`, `topic`, `tid`[a], and `frame` with known camera intrinsic matrix $\vec{K} \in R^{3\times4}$ and camera extrinsic matrix $[\vec{R}|\vec{t}]$, we can convert the position of an object in world coordinate, camera coordinate, or pixel coordinate by Equation (S1) and Equation (S3), where $\vec{R} \in R^{3\times3}$ and $\vec{t} \in R^{3\times1}$ indicate the rotation and translation matrices. $P_c \in R^{4\times1}$ is the 3D point position in camera coordinates and $P_w \in R^{4\times1}$ is the 3D point position in world coordinates. In both cases, they are represented in homogeneous coordinates. A camera extrinsic matrix $\vec{M} \in R^{4\times4}$ is used to denote a projective mapping from world coordinates to pixel coordinates.
>
> $$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \vec{K} \begin{bmatrix} \vec{R} & \vec{t} \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \vec{K} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \vec{P} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{S1}$$
>
> $$\vec{P} = \vec{K}\vec{M} \tag{S2}$$
>
> $$P_c = \vec{M}P_w \tag{S3}$$
>
> The elements of object's center position vector P_c can be acquired from the MOT ground truth data `x3d`, `y3d`, `z3d`, and `h3d`, and the corresponding P_w will be easily obtained by apply the inverse of frame-dependent matrix $\vec{M}$. In the x-z plane, arbitrary vehicle poses can be generated according to the primitive function, where r'_y is a unit tangent vector to the curve at (x'_w, z'_w) representing the target orientation of the car object. The trajectory curve and heading vectors are generated at the origin and then translated to the desired starting point. The curve was further rotated by θ using the rotation matrix $\mathcal{R}_\theta$ to align with the original trajectory's slope a, where a, θ are the slope and angle of original path, $\mathcal{R}$ is a corresponding rotation matrix. A detailed algorithm of the implementation of the primitive operation in 3D world coordinate is shown in the supplementary material.
>
> [a] A unique track identification number for each object instance in the scene.

Algorithm 1 Transformation algorithm used in cut-in.

Input: Input scene **S**, topic **T**, car object **tid**

Output: dataframe **df**

Let $t \in [t_s, t_e]$

$P_{c,t} \leftarrow \begin{bmatrix} x_{c,t} & y_{c,t} & z_{c,t} & 1 \end{bmatrix}$

$P_{w,t} \leftarrow \vec{M}^{-1} P_{c,t}$

Apply linear regression on $P_{w,t}$ to get slope a

Generate a curve by *primitive function* at origin $P^*_{w,t} \leftarrow \begin{bmatrix} x'_w & y'_w & z'_w & 1 \end{bmatrix}^w \; / \; r'_y$

Rotation matrix $\mathcal{R}_\theta$, where $\theta \leftarrow \arctan(a)$

for $t \leftarrow t_s$ to t_e **do**

    $\begin{bmatrix} x_{w,t} & z_{w,t} \end{bmatrix}^{\leftarrow} \mathcal{R}_\theta \begin{bmatrix} x'_{w,t} & z'_{w,t} \end{bmatrix}$

    $P_{c,t} \leftarrow \vec{M} P_{w,t}$

    $\vec{r} \leftarrow \mathcal{R}_\theta r'_y$

    $r_y \leftarrow (r_2 - z_c(t), r_1 - x_{c,t})$

end for

$x_c \leftarrow x_c + x_o$

$z_c \leftarrow z_c + z_o$

`df['x3d']` $\leftarrow x_c$

`df['y3d']` $\leftarrow y_c$

`df['z3d']` $\leftarrow z_c$

`df['ry']` $\leftarrow r_y$

return df

B Details of experiments

B.1 Primitive Generation

In this section, we provide more detailed pictures of primitive generation, mainly focusing on the visualization of trajectories in the world coordinate system and the camera coordinate system.

B.1.1 Exit Parallel Parking

In the world coordinate system, we generate the primitive function (blue curve) at the origin, push it to the target point, and convert to the camera coordinate system. The pink color in the figure is the original position of the vehicle parked on the side of the road in the camera coordinate system, while the dark blue point and the light blue arrow are the converted coordinate points and the vehicle head direction converted to the camera coordinate system according to the primitives, respectively.

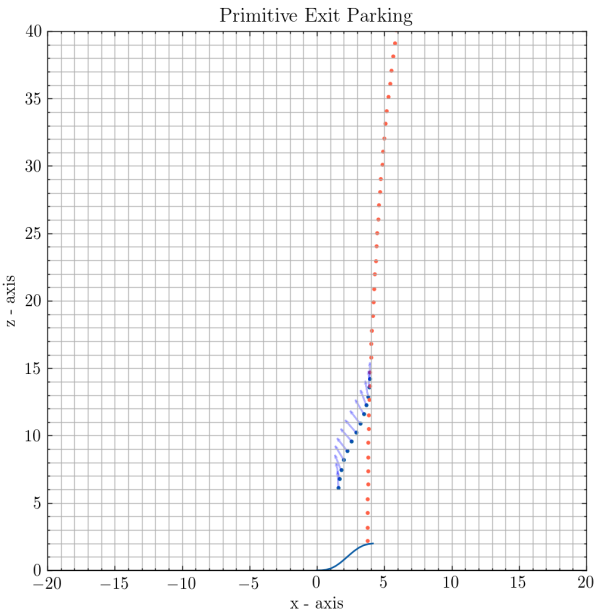


Figure S3: Visualization of primitive operation in camera space.

B.1.2 Cut-in Opposite

In the world coordinate system, we generate the primitive function (blue curve) at the origin and push it to the target starting point shown as the green curve. The pink color in the figure is the original path curve, while the dark blue point and light blue arrow are the rotated coordinate point and the direction of the converted car head, respectively. The following sections color markings are the same.

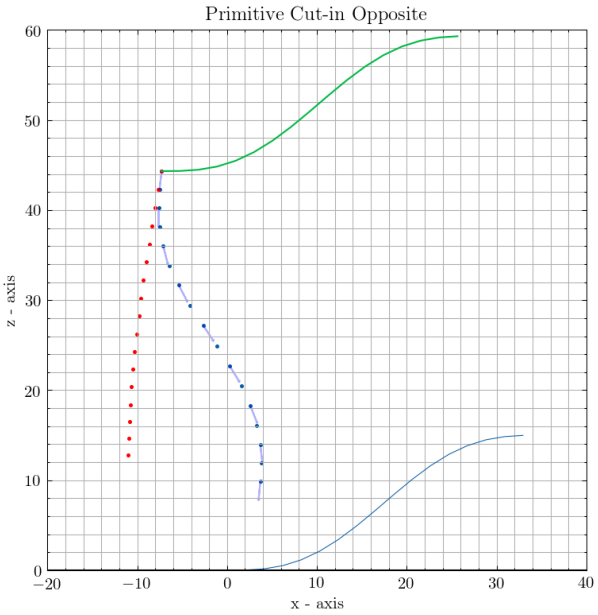


Figure S4: Visualization of primitive operation in world space.

B.1.3 Slalom

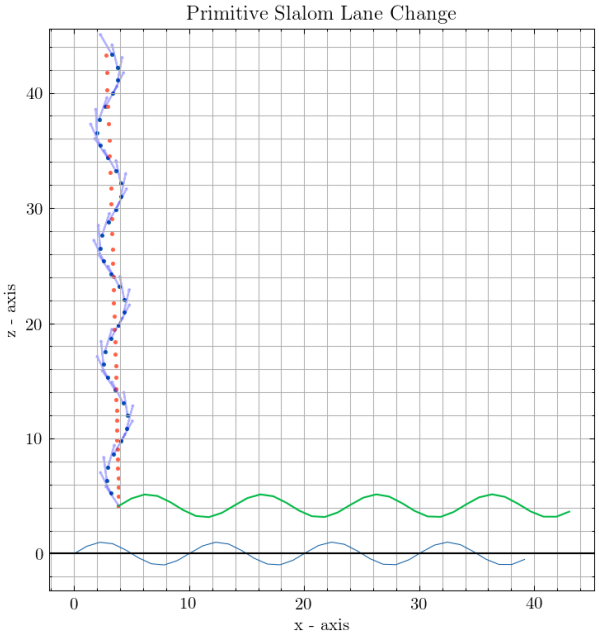


Figure S5: Visualization of slalom lane change operation in world space.

B.1.4 Cut-in

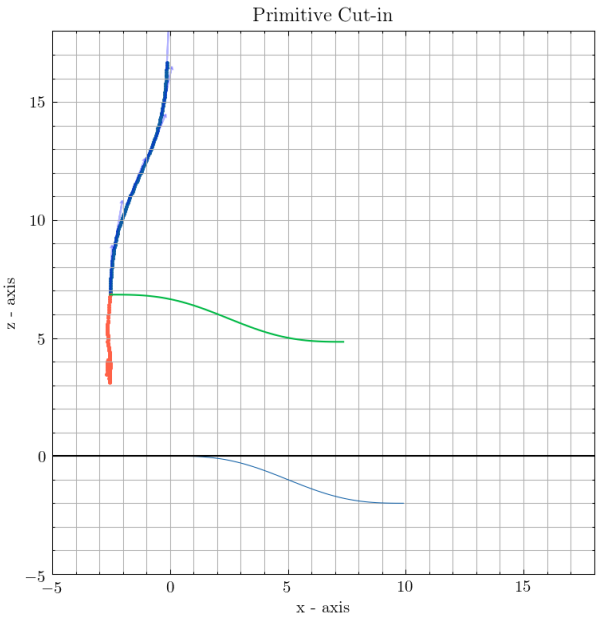


Figure S6: Visualization of cut-in operation in world space.

B.1.5 Brake

Unlike the above method, we simulate the real braking deceleration in the brake primitive and transform it into the velocity and displacement profile of the vehicle object by integration. By reprojecting them onto the original driving trajectory in the world coordinate system, they are then transformed into relative

position points in the camera coordinate system. Our approach is not just to bring the vehicle closer by a distance, but to pull the original vehicle to the real trajectory points based on the real physical travel.

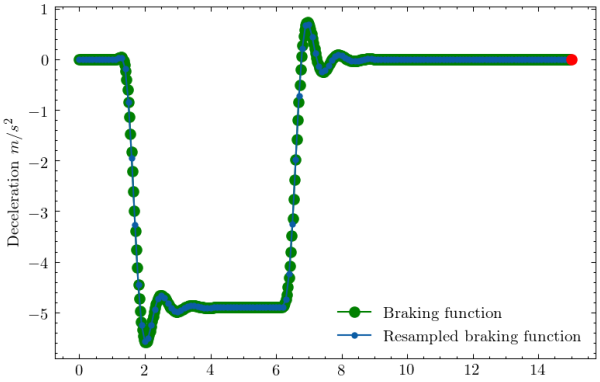


Figure S7: Braking acceleration based on the parameters input.

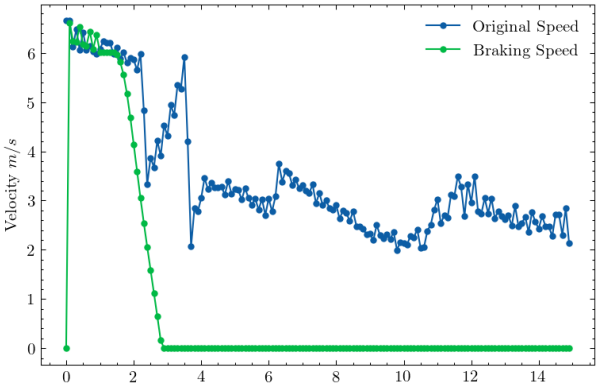


Figure S8: Original and modified speed profile.

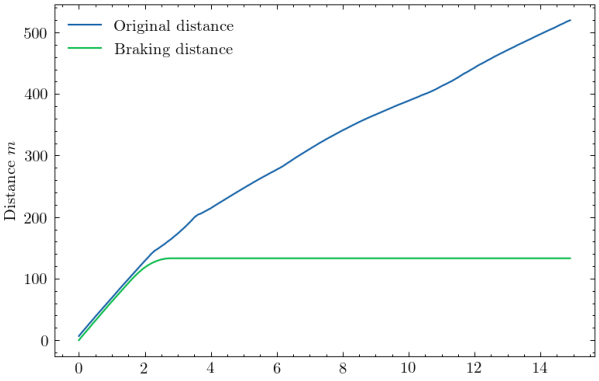


Figure S9: Original and modified travel distance profile

B.2 JSON file naming

We saved the generated JSON file by the following rule,

`<world>_<topic>_<tid>_<prim>_variable1#1_variable2#2.json`

where

- `<world>` is the name of a virtual world, which is the sequence number of the corresponding original "seed" real-world KITTI sequence (`0001`, `0002`, `0006`, `0018`, `0020`) in the 3D Object Detection Evaluation challenge [Geiger et al.].
- `<topic>` denotes one of the 10 different rendering variations in terms of imaging or weather conditions. `clone`: rendering as close as possible to original real-world KITTI conditions; `morning`: typical lighting conditions after dawn on a sunny day; `sunset`: lighting typical of slightly before sunset; `overcast`: typical overcast weather (diffuse shadows, strong ambient lighting).
- `tid` The operation will apply to which car object with number `'<tid>'`. Please look up the number under column tid in Table. S2.
- `prim` Primitive parameters defined in Table. S2, and corresponding value `variable`.

A sample JSON file that contains the (u, v), `zoom`, and r_y information is listed as follows:

```
[
    {
        "world": "0018",
        "topic": "clone",
        "source": "00140",
        "target": "00140",
        "operations": [
            {
                "type": "modify",
                "from": {
                    "u": "350.3",
                    "v": "287.0"
                },
                "to": {
                    "u": "247.9",
                    "v": "319.6",
                    "roi": [ 0, 0, 0, 0]
                },
                "zoom": "1.3252036238820235",
                "ry": "0.056247797876731065"
            }
        ]
    },
    ...
]
```

	"0001" Exit Parking (tid, x_e, y_e)	"0002" Cut-in Opposite (tid, f, A)	"0006" Slalom (tid, f, A)	"0018" Cut-in (tid, x_e, y_e)	"0020" Braking (tid, t_1, dt, μ)
Case 1	(63, 4, 1.8)	(0, 20, 9)	(7, 0.1, 0.5)	(1, 6, 4)	(6, 5, 2.5, 0.35)
Case 2	(63, 4, 2)	(0, 23, 9)	(7, 0.1, 1.0)	(1, 8, 4)	(6, 5, 2.5, 0.2)
Case 3	(63, 4, 2.2)	(0, 26, 9)	(7, 0.1, 1.5)	(1, 10, 4)	(6, 5, 2.5, 0.5)
Case 4	(63, 4.5, 1.8)	(0, 29, 9)	(7, 0.08, 0.5)	(1, 6, 3)	(6, 5, 4, 0.35)
Case 5	(63, 4.5, 2)	(0, 32, 9)	(7, 0.08, 1.0)	(1, 8, 3)	(6, 5, 4, 0.5)
Case 6	(63, 4.5, 2.2)	(0, 35, 9)	(7, 0.08, 1.5)	(1, 10, 3)	(6, 5, 4, 0.2)
Case 7	(63, 5, 1.8)	(0, 20, 12)	(7, 0.12, 0.5)	(1, 6, 5)	(6, 10, 2.5, 0.35)
Case 8	(63, 5, 2)	(0, 23, 12)	(7, 0.12, 1.0)	(1, 8, 5)	(6, 10, 2.5, 0.2)
Case 9	(63, 5, 2.2)	(0, 26, 12)	(7, 0.12, 1.5)	(1, 10, 5)	(6, 10, 2.5, 0.5)
Case 10	(70, 4, 1.8)	(0, 29, 12)	(2, 0.1, 0.5)	(2, 6, -4)	(6, 10, 4, 0.35)
Case 11	(70, 4, 2)	(0, 32, 12)	(2, 0.1, 1.0)	(2, 8, -4)	(6, 10, 4, 0.5)
Case 12	(70, 4, 2.2)	(0, 35, 12)	(2, 0.1, 1.5)	(2, 10, -4)	(6, 10, 4, 0.2)
Case 13	(70, 4.5, 1.8)	(0, 20, 15)	(2, 0.08, 0.5)	(2, 6, -3)	(6, 2, 2.5, 0.35)
Case 14	(70, 4.5, 2)	(0, 23, 15)	(2, 0.08, 1.0)	(2, 8, -3)	(6, 2, 2.5, 0.2)
Case 15	(70, 4.5, 2.2)	(0, 26, 15)	(2, 0.08, 1.5)	(2, 10, -3)	(6, 2, 2.5, 0.5)
Case 16	(70, 5, 1.8)	(0, 29, 15)	(2, 0.12, 0.5)	(2, 6, -5)	(6, 2, 4, 0.35)
Case 17	(70, 5, 2)	(0, 32, 15)	(2, 0.12, 1.0)	(2, 8, -5)	(6, 2, 4, 0.5)
Case 18	(70, 5, 2.2)	(0, 35, 15)	(2, 0.12, 1.5)	(2, 10, -5)	(6, 2, 4, 0.2)

Table S2: Parameters in primitives.

B.3 Class used in `MobileNet-V2`

No.	Class	No.	Class	No.	Class	No.	Class
1	Road	6	Pole	11	Sky	**16**	**Bus**
2	Sidewalk	7	Traffic ligh	**12**	**Person**	**17**	**Train**
3	Building	8	Traffic sign	**13**	**Rider**	**18**	**Motorcycle**
4	Wall	9	Vegetation	**14**	**Car**	**19**	**Bicycle**
5	Fence	10	Terrain	**15**	**Truck**		

Table S3

References

[Geiger et al.] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. 3d object detection evaluation 2017. `http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d`. (Accessed on 08/04/2022).

Glossary

A Amplitude of the a car in sine-wave slalom maneuver. Unit in m. S9, S10

μ The constantly steady braking acceleration we designed to reach. Unit in $\#g$. For example, $0.5g = 4.9m/s^2$. S9, S10

dt The duration of applying a braking. $d_t = t_3 - t_2$. Unit in s. S9, S10

f Frequency of the a car in sine-wave slalom maneuver. Unit in Hz. S9, S10

r_y KITTI-like 3D object `rotation about y`, rotation around Y-axis (yaw) in camera coordinates $[-\pi..\pi]$ (KITTI convention is $r_y == 0$ iff object is aligned with x-axis and pointing right). S10

t_1 Starting time of applying a braking. Unit in s. S9, S10

x_e Longitudinal movement of a car in cut-in. S9, S10

y_e Lateral movement of a car in cut-in. S9, S10

delete A description used in JSON. Delete an object from current view by 3D-SDN. S10

frame Frame index in the video (starts from 0). S10

modify A description used in JSON. Moving an object from a position to another by 3D-SDN. S10

motgt Multi-Object Tracking Ground Truth. S10

scene Scene shares the **world** definition in vKITTI. It's the name of a virtual world, which is the sequence number of the corresponding original "seed" real-world KITTI sequence (0001, 0002, 0006, 0018, 0020). S10

tid Track identification number (unique for each object instance). In the context of DANGER, tid means the id of a specific car object. S9, S10

topic Denotes one of the 10 different rendering variations in terms of imaging or weather conditions: **clone**: rendering as close as possible to original real-world KITTI conditions; **15-deg-right**: horizontal rotation of the camera 15 degrees to the right; **15-deg-left**: horizontal rotation of the camera 15 degrees to the left; **30-deg-right**: horizontal rotation of the camera 30 degrees to the right;

30-deg-left: horizontal rotation of the camera 30 degrees to the left; **morning**: typical lighting conditions after dawn on a sunny day; **sunset**: lighting typical of slightly before sunset; **overcast**: typical overcast weather (diffuse shadows, strong ambient lighting); **fog**: fog effect implemented via a volumetric formula; **rain**: simple rain particle effect (ignoring the refraction of water drops on the camera). Only inference **CLONE** in this implementation. S10

w3d, h3d, l3d KITTI-like 3D object 'dimensions', respectively width, height, length in meters. S10

x3d, y3d, z3d KITTI-like 3D object 'location', respectively x, y, z in camera coordinates in meters (center of bottom face of 3D bounding box). S10

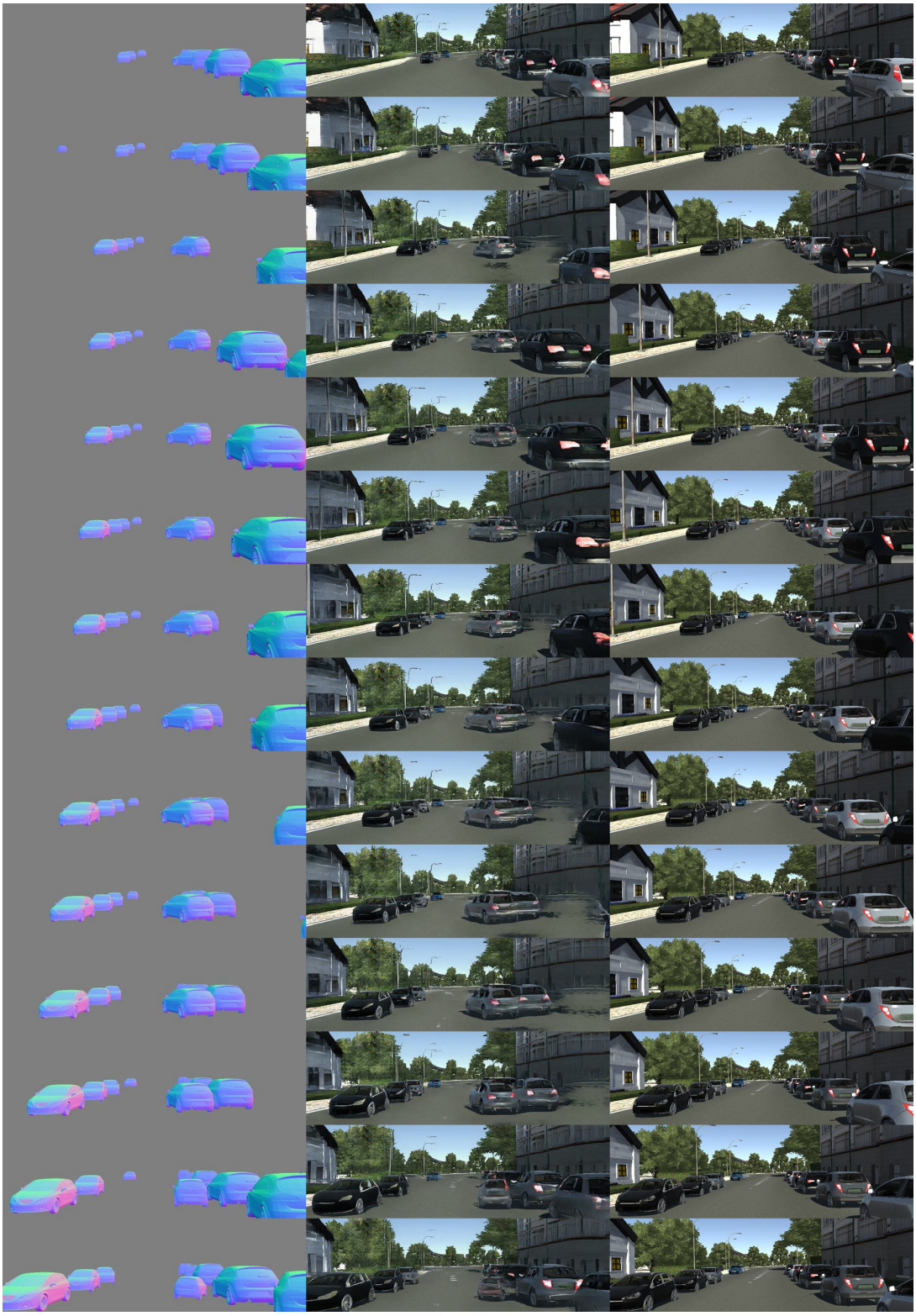


Figure S10: `0001_clone_tid70_epp_xe5_ye2.2`

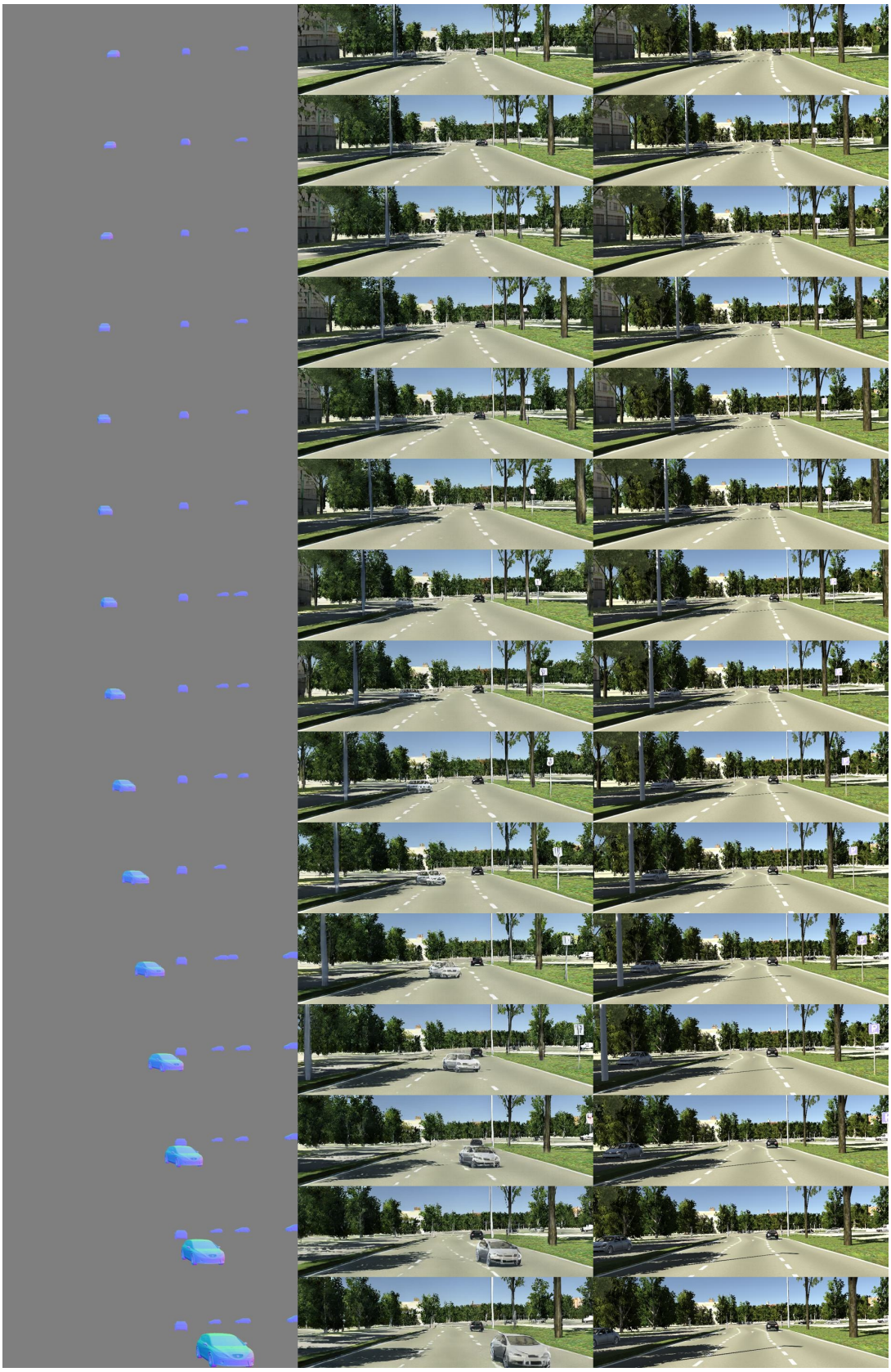


Figure S11: `0002_clone_tid0_cio_xe40_ye15`

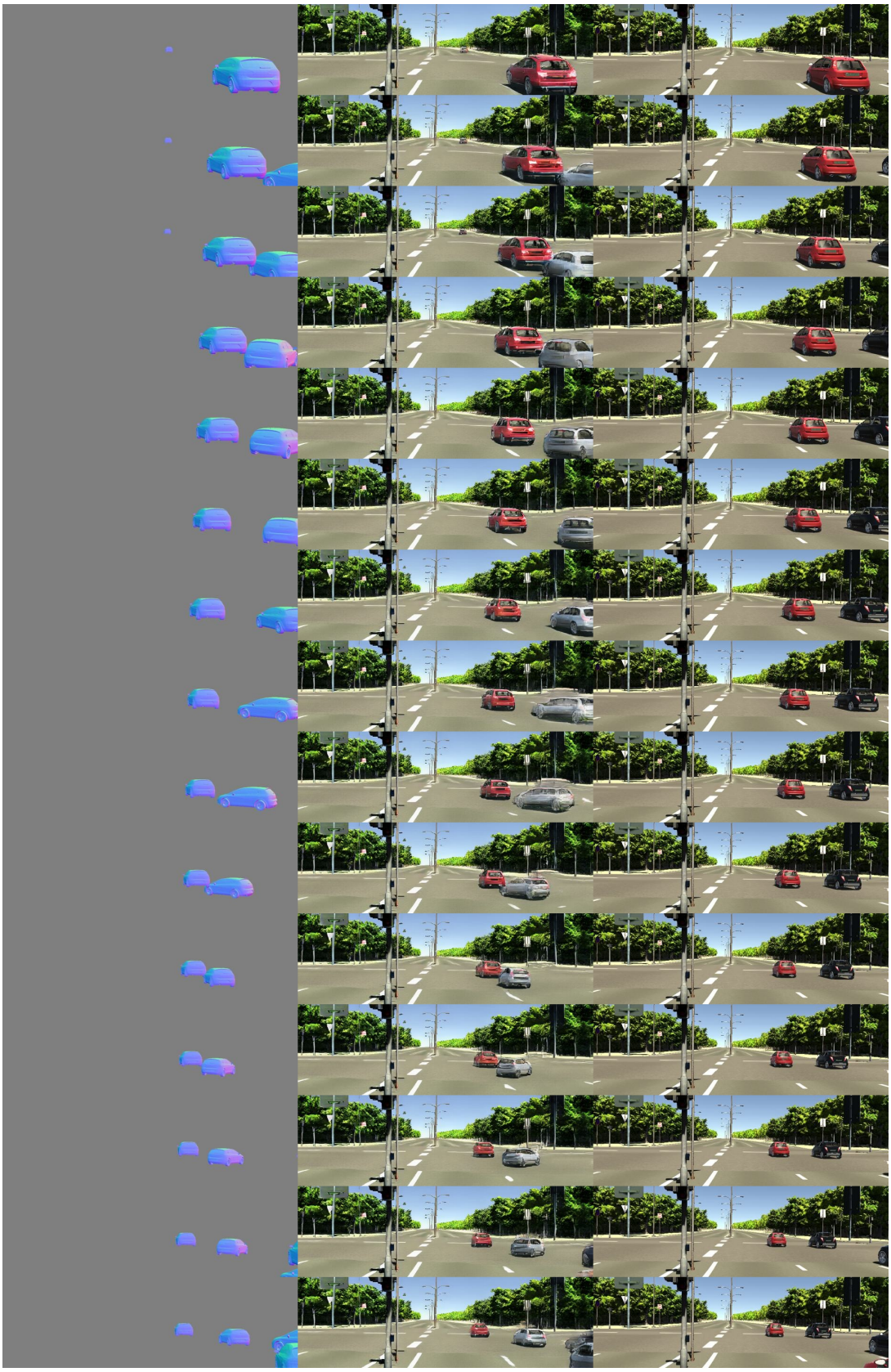


Figure S12: `0006_clone_tid7_lc_f0pt12_A1pt5`

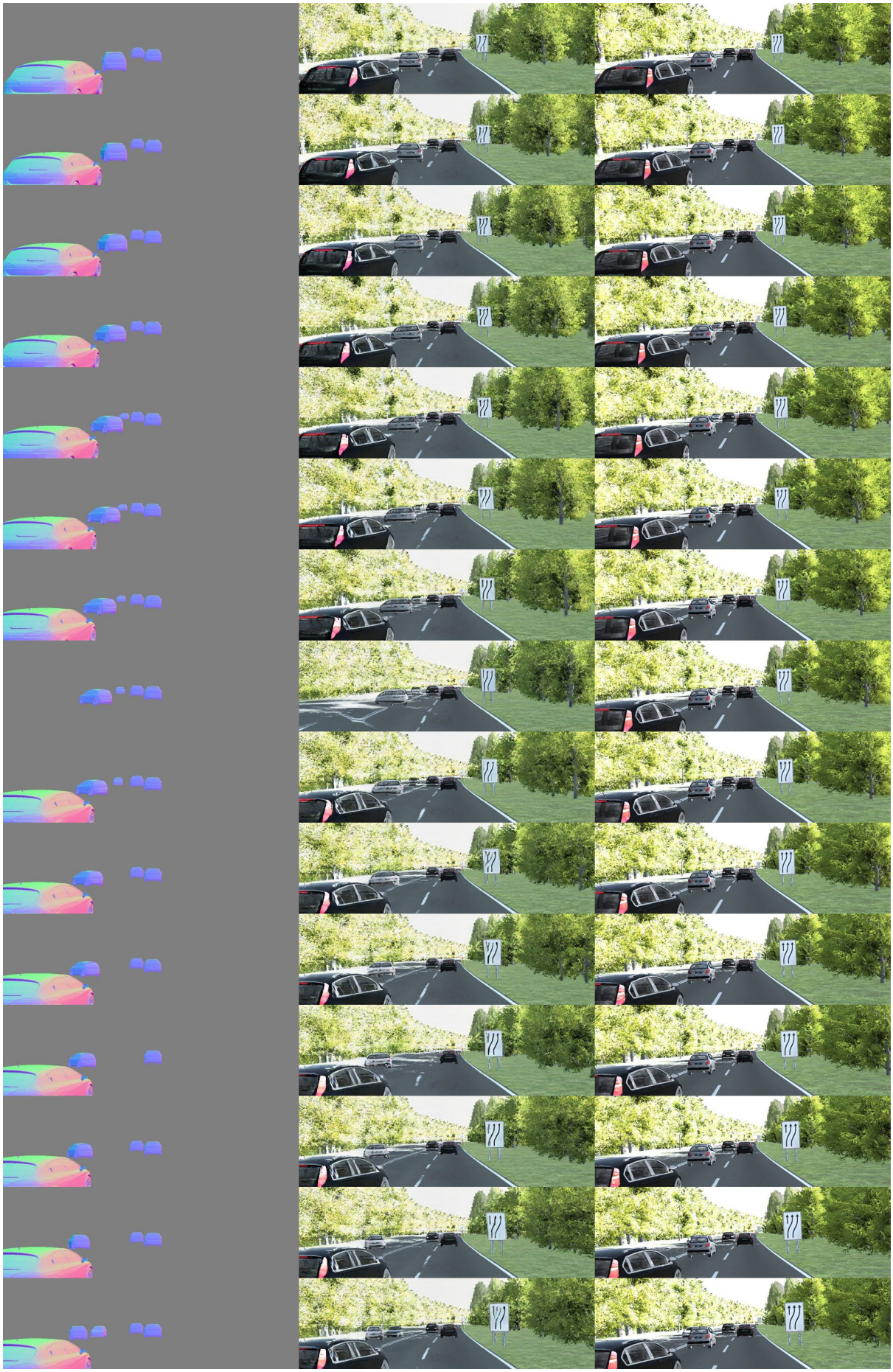


Figure S13: 0018_clone_tid2_ci_xe10_ye5

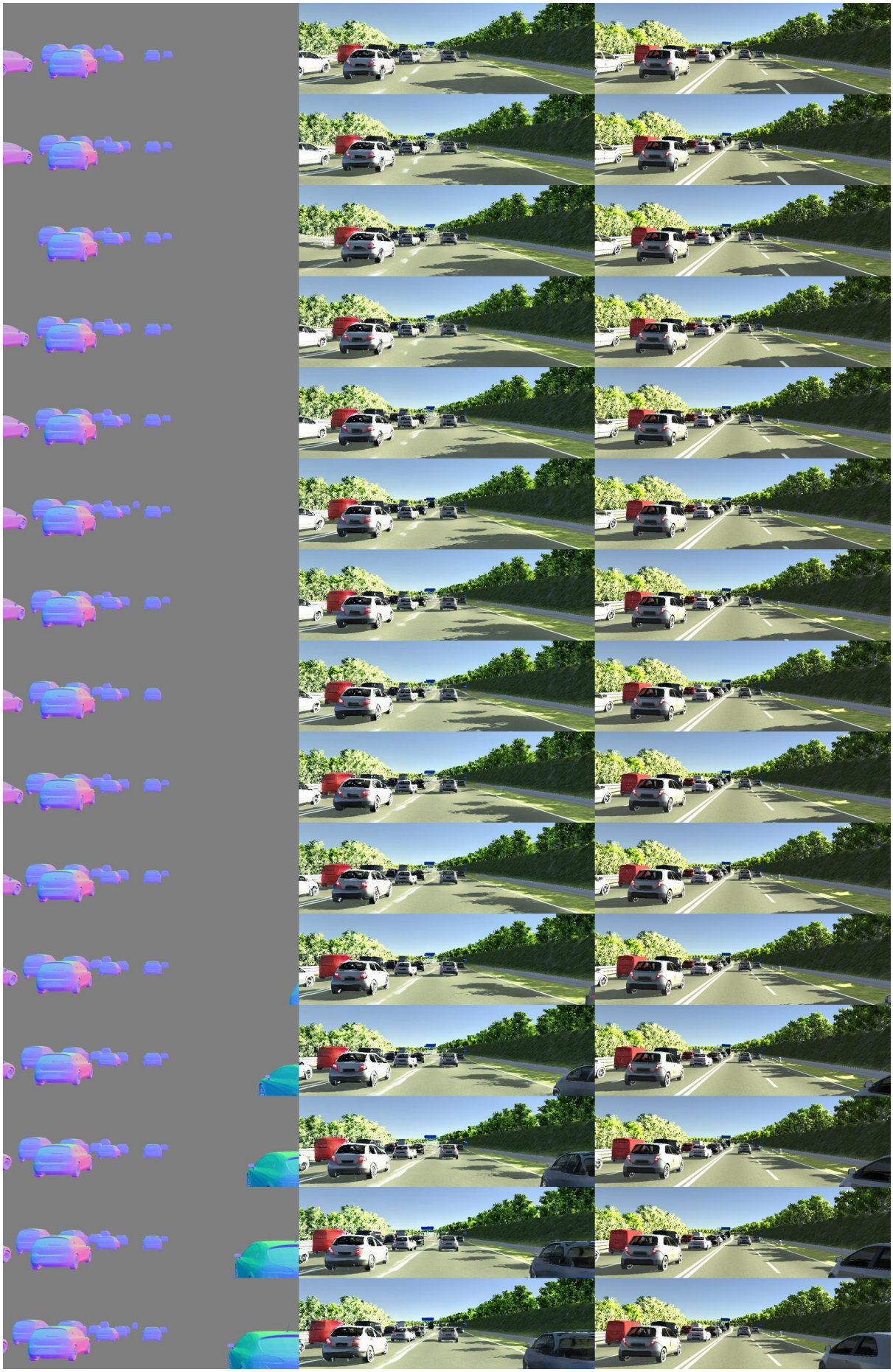


Figure S14: 0020_clone_tid16_br_t010_dt4_g0pt35