Synthehicle: Multi-Vehicle Multi-Camera Tracking in Virtual Cities

Fabian Herzog  Junpeng Chen  Torben Teepe  Johannes Gilg
Stefan Hörmann  Gerhard Rigoll
Technical University of Munich, Germany

Abstract

Smart City applications such as intelligent traffic routing, accident prevention or vehicle surveillance rely on computer vision methods for exact vehicle localization and tracking. Privacy issues make collecting real data difficult, and labeling data is a time-consuming and costly process. Due to the scarcity of accurately labeled data, detecting and tracking vehicles in 3D from multiple cameras proves challenging to explore. We present a massive synthetic dataset for multiple vehicle tracking and segmentation in multiple overlapping and non-overlapping camera views. Unlike existing datasets, which only provide tracking ground truth for 2D bounding boxes, our dataset additionally contains perfect labels for 3D bounding boxes in camera- and world coordinates, depth estimation, and instance, semantic and panoptic segmentation. The dataset consists of 17 hours of labeled video material, recorded from 340 cameras in 64 diverse day, rain, dawn, and night scenes, making it the most extensive dataset for multi-target multi-camera tracking so far. We provide baselines for detection, vehicle re-identification, and single- and multi-camera tracking. Code and data are publicly available.¹

1. Introduction

As cities grow larger in population, increased car traffic causes jams, pollution, and accidents. Future smart cities, where multiple sensors (e.g., RGB cameras) are placed near crossroads, could reduce these problems through intelligent traffic management driven by computer vision.

In particular, multi-target multi-camera tracking (MTMCT) is an essential task for such methods as it enables 3D localization and provides information for scene understanding. In MTMCT, distinct objects must be unambiguously tracked across multiple cameras through space and time. The tracking problem is generally challenging, even in the single-camera case, primarily due to occlusions. Information added by multiple cameras can be beneficial but complicates data processing. Existing solutions for MTMCT [52, 24, 48, 39, 44] are mainly based on tracking-by-detection, in which an object detector is applied to obtain frame-wise sets of 2D bounding boxes which are then processed by a deep neural feature extractor for data association. Recently, the single-camera tracking community has started paying more attention to more complex tasks, such as 3D multi-object tracking [63, 69] and tracking and segmentation [59, 62]. Data annotation, particularly for multi-camera setups and 3D localization and segmentation, is time-consuming and expensive. To this day, there is no real dataset for multi-target multi-camera tracking with 3D and segmentation annotations. Both enable potentially better 3D localization than approaches based on 2D bounding boxes.

We address the scarcity of 3D and segmentation data and introduce Synthehicle, a massive synthetic dataset for multiple vehicle detection, tracking, and segmentation across multiple cameras with overlapping and non-overlapping field of views (see Figure 1 for example annotations). The dataset consists of 17 hours of labeled video material,

¹Code and data: https://github.com/fubel/synthehicle
recorded from 340 cameras placed around crossroads and highways in 64 diverse day, rain, dawn and night scenes, and has been created using CARLA [12]. Summarized, our contributions are:

- a massive synthetic dataset for multi-target multi-camera tracking with 2D, 3D, segmentation and depth annotations;
- a public evaluation server to test methods against ground truth of the test sets; and
- baseline results for 2D detection, vehicle re-identification, single- and multi-camera multi-vehicle tracking tasks.

Code for data generation, detection, re-identification and tracking, as well as all the generated data and evaluation scripts are publicly available.

2. Related Work

Vehicle Detection and Tracking Datasets CityFlow [56, 46] is the established dataset to train and test vision methods for smart city applications such as multi-camera vehicle tracking, vehicle re-identification, and vehicle counting. However, it does not include 3D box annotations, depth maps, or segmentations. KITTI [19, 18] is loosely related as it includes data for 2D and 3D detection and tracking of vehicles. As KITTI aims at improving autonomous vehicles, the data is captured from specific ego-motion camera angles. nuScenes [4], also designed for autonomous driving, is similar to KITTI, but significantly larger. It provides more ego-motion camera angles and includes night and rain scenarios. Waymo Open [55] is similar in design but even more extensive than nuScenes. Neither KITTI, nuScenes nor Waymo Open provide the data for smart city scenarios. They focus on ego-motion and stereo vision by design and are unsuitable for studying problems such as vehicle re-identification and multi-vehicle multi-camera tracking.

Vehicle Re-Identification Datasets Just like person tracking relies on person feature extraction, most multi-vehicle trackers build upon vehicle re-identification methods [48, 39, 57, 54, 67, 32, 66], which are usually trained on the VeRi-776 [36, 34, 37] or CityFlow re-identification [56] datasets. Compared to VeRi-776, CityFlow was recorded in more diverse scenarios and viewing angles. Because trackers have to deal with occlusions and bad lighting conditions, the VERI-Wild vehicle re-identification dataset [38] has been proposed to provide images and annotations recorded in the wild.

Synthetic Datasets Recording and labeling real data is time-consuming and expensive. In addition, data protection rights are potentially infringed when recording humans. Therefore, numerous synthetic datasets have been published to solve common computer vision tasks, such as segmentation [58, 20, 53, 27, 49, 31], detection [40, 1], pose estimation [14], and tracking [14, 30, 13, 16, 10, 26]. In person tracking, the most significant synthetic datasets are JTA [14], which presents a dataset for single-camera pose-tracking based on the video game GTA V. Closely related, MTA [30] offers multi-camera multi-person tracking data that can be seen as a replacement for the DukeMTMC dataset. MOTSynth showed that training on synthetic data can improve tracking results on real data [13]. The synthetic VehicleX [68] dataset was generated for vehicle re-identification. Related datasets also based on CARLA are KITTI-CARLA [10], which provides synthetic lidar data analogous to KITTI [19], V2I-CARLA [61] for synthetic vehicle re-identification, and the Paris-CARLA-3D [11] dataset for 3D mapping.

3. Dataset Analysis

3.1. Overview and Comparison

So far, the MTMC vehicle tracking community has focused on tracking-by-detection, a methodology based on 2D object detections and subsequent appearance feature extraction [48, 39, 57, 54, 67, 32, 66]. Unlike other datasets, which are designed for tracking targets enclosed by 2D bounding boxes, Synthehicle provides 3D annotations in terms of 3D bounding boxes and world coordinates, segmentations (instance, semantic and panoptic), and depth images. In Table 1, we compare Synthehicle to established tracking datasets, such as PETS09, WILDTRACK and CityFlow. Synthehicle is the only dataset with all of the listed annotation types, and the longest dataset in duration.
In Synthehicle, scenes are rendered under four different ambience configurations: day (a), dawn (b), rain (c), and night (d).

### Table 1: Overview of existing MTMCT datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Cams</th>
<th># Boxes</th>
<th># Scenes</th>
<th>Density</th>
<th>Duration</th>
<th>Targets</th>
<th>3D Boxes</th>
<th>Depth</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DukeMTMC (offline) [51]</td>
<td>8</td>
<td>4077132</td>
<td>1</td>
<td>1.9</td>
<td>11.33h</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>CAMPUS [65]</td>
<td>16</td>
<td>12264</td>
<td>4</td>
<td>–</td>
<td>45m</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>EPFL-RLC</td>
<td>3</td>
<td>6132</td>
<td>1</td>
<td>–</td>
<td>6.6 min</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>PETS09</td>
<td>7</td>
<td>4650</td>
<td>8</td>
<td>–</td>
<td>2 min</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>WILDTRACK [5]</td>
<td>7</td>
<td>56000</td>
<td>1</td>
<td>23.8</td>
<td>1h</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>CityFlow [56]</td>
<td>46</td>
<td>313931</td>
<td>6</td>
<td>2.5</td>
<td>3.58h</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Synth.</td>
<td>1*</td>
<td>10000000</td>
<td>512</td>
<td>20</td>
<td>4.27h</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>JTA [14]</td>
<td>1*</td>
<td>40780800</td>
<td>768</td>
<td>29.5</td>
<td>15.36h</td>
<td>Person</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MOTSynth [13]</td>
<td>1*</td>
<td>37324348</td>
<td>1</td>
<td>24.8</td>
<td>10.2h</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>MTA [30]</td>
<td>6</td>
<td>37324348</td>
<td>1</td>
<td>24.8</td>
<td>10.2h</td>
<td>Person</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Synthehicle (ours)</td>
<td>340</td>
<td>4623184</td>
<td>64</td>
<td>7.45</td>
<td>17.00h</td>
<td>Vehicles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1 shows a comparison of Synthehicle to related datasets. A ✓ indicates whether 3D boxes, depth annotations, and segmentations are included in the ground truth, respectively. For multi-camera datasets, the number of cameras corresponds to the total number of videos in the dataset. Naturally, single-camera datasets have one camera per scene, and their camera numbers are marked with *. The last row compares the individual statistics of our dataset directly to CityFlow.

Compared to the closest related real dataset, CityFlow, Synthehicle is three times denser in terms of average number of vehicles per frame, has more than ten times as many scenes and fourteen times as many annotated bounding boxes. It also surpasses all other multi-camera tracking datasets in the number of cameras. Table 2 provides a detailed list of all Synthehicle scenes and their classification into train and test split.

### 3.2. Data Recording

CARLA [12] is an open-source simulation tool for building urban traffic scenarios and has been successfully employed to generate numerous synthetic datasets for computer vision tasks [58, 20, 53, 27, 49, 31, 40, 1, 14, 30, 13, 16, 10, 26]. We utilize CARLA’s rich set of realistic simulated sensors to render urban vehicle tracking scenarios using RGB, depth, and semantic LIDAR sensors in CARLA’s eight pre-designed town maps. The data recording process can be described as follows.

First, we define two scenes for each of the eight towns - one for an overlapping camera view setup (O) and the other for a non-overlapping setup (N). We place a varying number of cameras (3 to 8) in each scene. The cameras are placed such that the vehicles are viewed from a high angle to mimic real-world positions, e.g., on top of traffic lights, similar to CityFlow [56]. After defining the camera networks, we randomly spawn vehicles and pedestrians. The maximum number of vehicles that could spawn on a map is set to 200. Pedestrians mainly fulfill the task of influencing the otherwise monotonous flow of traffic. Models and appearances for vehicles and pedestrians are randomly chosen from CARLA’s model pool and a list of realistic vehicle colors, matching real-life urban scenarios (cf. Figure 4).

Vehicle routing and rules are controlled by CARLA’s TrafficManager, which was designed to manage vehicles in autopilot mode. Some vehicles will disobey traffic rules by driving too fast or crossing red traffic lights, providing more diverse and less predictable trajectories. For every defined camera, we deploy an RGB sensor to record the traffic scenes with a resolution of $1920 \times 1080$ px and a frame rate of 10 fps. Semantic rotating LIDAR sensors capture information about all vehicles in a camera’s field of view. The sensor is implemented using ray-casting and exposes all information about objects hit by a ray. We use the sensor to filter out annotations for heavily occluded objects, e.g., for cars hidden behind buildings. We attach two different seg-
Cameras calibrations, 2D and 3D annotations while recording the scenes.

Finally, we extract perfect ground truth annotations while the vehicle provides a vast variety of traffic flows and vehicle trajectories. Figure 2 shows an example frame recorded from the same camera under different configurations: Day, dawn, rain, and night.

To record semantic and instance segmentations and to obtain annotation sensors and a depth sensor to each RGB camera to record semantic and instance segmentations and to obtain depth information. The recording of all sensors lasts 1800 frames. Scenes are recorded as described above in four different ambience configurations: Day, dawn, rain, and night. Figure 2 shows an example frame recorded from the same camera under different configurations. When recording a scene for a weather configuration, all vehicle and pedestrian spawns and traffic flows will be randomized - only the sensor placement is constant for each scene. This way, Synthehicle provides a vast variety of traffic flows and vehicle trajectories. Finally, we extract perfect ground truth annotations while recording the scenes.

### 3.3. Data Types

We use CARLA's Python API to extract a variety of ground truth annotations: Camera calibrations, 2D and 3D detections, semantic, instance and panoptic segmentations, depth information and multi-camera tracking ground truth.

### 3.4. Calibrations

For each camera, we obtain its $3 \times 3$ camera intrinsic matrix $K$ and its $4 \times 4$ world-to-camera matrix $M_{w2c} = [R, t]$, which is the inverse of the camera extrinsic matrix. These matrices can be used for converting world points to image points and vice versa. We also provide $(x, y, z)$ 3D world positions for all cameras and their pitch, roll and yaw.

### 3.5. 3D Detections

For each object in the scene, we obtain its 3D bounding box in world $(x, y, z)$-coordinates directly from CARLA. The oriented bounding box is defined by eight corner points and yaw rotation. Each world point $(x_w, y_w, z_w)^T \in \mathbb{R}^3$ is projected to camera coordinates using the $4 \times 4$ world-to-camera matrix $M_{w2c}$ to obtain the point in image coordinates $(x, y)^T$ via

$$
\begin{bmatrix}
\tilde{x}_1 \\
\tilde{y}_1 \\
\tilde{z}_1
\end{bmatrix} = M_{w2c}
\begin{bmatrix}
x_w \\
y_w \\
z_w
\end{bmatrix},
\begin{bmatrix}
x_1 \\
y_1 \\
z_1
\end{bmatrix} = K
\begin{bmatrix}
\tilde{y}_1 \\
\tilde{z}_1
\end{bmatrix},$$

Table 2: Overview over all 64 Synthehicle scenes. Scenes have a frame rate of 10fps, a resolution of 1920x1080 and a duration of 1800 frames per camera. The dataset is split into train and test scenes. The markers -O- and -N- in the scene names indicate overlapping and non-overlapping camera topology, respectively.

<table>
<thead>
<tr>
<th>Scene</th>
<th># Cams</th>
<th># Boxes</th>
<th># Tracks</th>
<th>Density</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Town01-O-dawn</td>
<td>4</td>
<td>48048</td>
<td>82</td>
<td>6.67</td>
<td></td>
</tr>
<tr>
<td>Town01-O-day</td>
<td>4</td>
<td>57539</td>
<td>80</td>
<td>7.99</td>
<td></td>
</tr>
<tr>
<td>Town01-O-night</td>
<td>4</td>
<td>51725</td>
<td>73</td>
<td>7.18</td>
<td></td>
</tr>
<tr>
<td>Town01-O-rain</td>
<td>4</td>
<td>70595</td>
<td>69</td>
<td>9.80</td>
<td></td>
</tr>
<tr>
<td>Town02-O-dawn</td>
<td>3</td>
<td>61337</td>
<td>69</td>
<td>11.35</td>
<td></td>
</tr>
<tr>
<td>Town02-O-day</td>
<td>3</td>
<td>64493</td>
<td>74</td>
<td>11.86</td>
<td></td>
</tr>
<tr>
<td>Town02-O-night</td>
<td>3</td>
<td>47619</td>
<td>64</td>
<td>8.81</td>
<td></td>
</tr>
<tr>
<td>Town02-O-rain</td>
<td>3</td>
<td>43416</td>
<td>72</td>
<td>8.04</td>
<td></td>
</tr>
<tr>
<td>Town03-O-dawn</td>
<td>8</td>
<td>200088</td>
<td>126</td>
<td>13.89</td>
<td></td>
</tr>
<tr>
<td>Town03-O-day</td>
<td>8</td>
<td>192994</td>
<td>124</td>
<td>13.40</td>
<td></td>
</tr>
<tr>
<td>Town03-O-night</td>
<td>8</td>
<td>207035</td>
<td>117</td>
<td>14.37</td>
<td></td>
</tr>
<tr>
<td>Town03-O-rain</td>
<td>8</td>
<td>192072</td>
<td>115</td>
<td>13.33</td>
<td></td>
</tr>
<tr>
<td>Town04-O-dawn</td>
<td>4</td>
<td>33644</td>
<td>39</td>
<td>4.67</td>
<td></td>
</tr>
<tr>
<td>Town04-O-day</td>
<td>4</td>
<td>35816</td>
<td>47</td>
<td>4.97</td>
<td></td>
</tr>
<tr>
<td>Town04-O-night</td>
<td>4</td>
<td>28173</td>
<td>44</td>
<td>3.91</td>
<td></td>
</tr>
<tr>
<td>Town04-O-rain</td>
<td>4</td>
<td>23086</td>
<td>42</td>
<td>3.20</td>
<td></td>
</tr>
<tr>
<td>Town05-O-dawn</td>
<td>6</td>
<td>131720</td>
<td>96</td>
<td>12.19</td>
<td></td>
</tr>
<tr>
<td>Town05-O-day</td>
<td>6</td>
<td>113413</td>
<td>96</td>
<td>10.50</td>
<td></td>
</tr>
<tr>
<td>Town05-O-night</td>
<td>6</td>
<td>107870</td>
<td>88</td>
<td>9.98</td>
<td></td>
</tr>
<tr>
<td>Town05-O-rain</td>
<td>6</td>
<td>127385</td>
<td>96</td>
<td>11.79</td>
<td></td>
</tr>
<tr>
<td>Town06-O-dawn</td>
<td>4</td>
<td>19734</td>
<td>45</td>
<td>2.74</td>
<td></td>
</tr>
<tr>
<td>Town06-O-day</td>
<td>4</td>
<td>13918</td>
<td>55</td>
<td>1.93</td>
<td></td>
</tr>
<tr>
<td>Town06-O-night</td>
<td>4</td>
<td>15859</td>
<td>45</td>
<td>2.20</td>
<td></td>
</tr>
<tr>
<td>Town06-O-rain</td>
<td>4</td>
<td>19308</td>
<td>57</td>
<td>2.68</td>
<td></td>
</tr>
<tr>
<td>Town07-O-dawn</td>
<td>4</td>
<td>54679</td>
<td>58</td>
<td>7.59</td>
<td></td>
</tr>
<tr>
<td>Town07-O-day</td>
<td>4</td>
<td>56314</td>
<td>57</td>
<td>7.82</td>
<td></td>
</tr>
<tr>
<td>Town07-O-night</td>
<td>4</td>
<td>80797</td>
<td>46</td>
<td>11.22</td>
<td></td>
</tr>
<tr>
<td>Town07-O-rain</td>
<td>4</td>
<td>46584</td>
<td>59</td>
<td>6.47</td>
<td></td>
</tr>
<tr>
<td>Town10HD-O-dawn</td>
<td>5</td>
<td>95426</td>
<td>110</td>
<td>10.60</td>
<td></td>
</tr>
<tr>
<td>Town10HD-O-day</td>
<td>5</td>
<td>165259</td>
<td>116</td>
<td>18.36</td>
<td></td>
</tr>
<tr>
<td>Town10HD-O-night</td>
<td>5</td>
<td>89170</td>
<td>100</td>
<td>9.90</td>
<td></td>
</tr>
<tr>
<td>Town10HD-O-rain</td>
<td>5</td>
<td>98855</td>
<td>125</td>
<td>10.98</td>
<td></td>
</tr>
</tbody>
</table>
where $K \in \mathbb{R}^{3 \times 3}$ denotes the camera intrinsic matrix and $(x, y)^T = (x_1/z_1, y_1/z_1)^T$. Note that CARLA uses UnrealEngine’s left-handed $z$-up coordinate system, which is why the axes have to be permuted and the $z$-axis reversed before multiplying with $K$ to obtain $(x, y)^T$ in image space. We store 3D boxes in COCO-format [33] as $\{(x_i, y_i, z_i, w_i, h_i, l_i, \theta_i)\}_i$, where $(x_i, y_i, z_i)$ is the 3D center point, $(w_i, h_i, l_i)$ describe weight, height and length of the box, and $\theta_i$ is the yaw rotation.

### 2D Detections

The 2D detections are obtained in image coordinates from the projected 3D detections. If $B_{3D} = \{c_1, \ldots, c_8\}$ is the 3D bounding box with its eight corner points, we obtain $(x_1, y_1, x_2, y_2)^T = (\min_x B_{3D}, \min_y B_{3D}, \max_x B_{3D}, \max_y B_{3D})^T$, i.e., by choosing the 2D box as the smallest rectangle containing all projected 3D vertices. Since 2D boxes acquired from this min-max-projection are usually larger than the enclosed target object, we tighten the box using the objects semantic segmentation label (see Figure 3). Bounding boxes smaller than $32 \times 32$ are filtered out from 2D detections, but are still kept in our general annotations.

### Segmentation

CARLA provides two sensors for segmentation: A semantic segmentation and an instance segmentation camera. We capture images from both cameras for each frame and directly store the corresponding output images. Having obtained semantic and instance segmentation labels, the panoptic pixel labels can easily be calculated by looking up the corresponding segmentation pixel for each instance segmentation pixel. Figure 1 illustrates an example of semantic and instance segmentation annotations.

### Depth Buffer

Leveraging CARLA’s depth sensor we also capture the depth map for each frame. The depth information can be used to improve camera projections since it provides scaling in depth dimension. Figure 1 includes an example.

### Tracking

We store the tracking ground truth for 2D and 3D boxes in COCO and MOTChallenge [41] format. Unlike CityFlow [56], our ground truth contains boxes for both vehicles and pedestrians, and boxes are included also if they are only visible in one camera.

### 4. Experiments

We conduct the following experiments to show the performance of existing methods on the Synthehicle dataset and to provide baselines for future research: 2D vehicle detection (Section 4.1), vehicle re-identification (Section 4.2), single-camera multi-vehicle tracking (Section 4.3), and multi-vehicle multi-camera tracking (Subsection 4.4).

#### 4.1. 2D Vehicle Detection

An object detector processes an image $I \in \mathbb{R}^{W \times H \times 3}$ and returns set of distinct objects $\{(x_i, y_i, w_i, h_i, c_i, s_i)\}_i$ with localization coordinates.

![Figure 3: Bounding Boxes in Synthehicle.](image)

![Figure 4: Vehicle colors in CARLA.](image)

<table>
<thead>
<tr>
<th>Train</th>
<th>Test</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_{5}$</th>
<th>AP$_{M}$</th>
<th>AP$_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>0.242</td>
<td>0.438</td>
<td>0.217</td>
<td>0.003</td>
<td>0.149</td>
<td>0.460</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>0.597</td>
<td>0.842</td>
<td>0.651</td>
<td>0.151</td>
<td>0.480</td>
<td>0.785</td>
</tr>
<tr>
<td>Day</td>
<td>Day</td>
<td>0.587</td>
<td>0.842</td>
<td>0.652</td>
<td>0.082</td>
<td>0.469</td>
<td>0.769</td>
</tr>
<tr>
<td>Dawn</td>
<td>Dawn</td>
<td>0.608</td>
<td>0.866</td>
<td>0.660</td>
<td>0.139</td>
<td>0.505</td>
<td>0.792</td>
</tr>
<tr>
<td>Rain</td>
<td>Rain</td>
<td>0.568</td>
<td>0.822</td>
<td>0.611</td>
<td>0.122</td>
<td>0.447</td>
<td>0.791</td>
</tr>
<tr>
<td>Night</td>
<td>Night</td>
<td>0.506</td>
<td>0.780</td>
<td>0.540</td>
<td>0.119</td>
<td>0.381</td>
<td>0.666</td>
</tr>
<tr>
<td>All</td>
<td>Day</td>
<td>0.626</td>
<td>0.870</td>
<td>0.693</td>
<td>0.116</td>
<td>0.510</td>
<td>0.808</td>
</tr>
<tr>
<td>All</td>
<td>Dawn</td>
<td>0.640</td>
<td>0.882</td>
<td>0.701</td>
<td>0.182</td>
<td>0.533</td>
<td>0.827</td>
</tr>
<tr>
<td>All</td>
<td>Rain</td>
<td>0.597</td>
<td>0.840</td>
<td>0.648</td>
<td>0.162</td>
<td>0.476</td>
<td>0.818</td>
</tr>
<tr>
<td>All</td>
<td>Night</td>
<td>0.522</td>
<td>0.777</td>
<td>0.560</td>
<td>0.171</td>
<td>0.383</td>
<td>0.696</td>
</tr>
</tbody>
</table>

Table 3: 2D detection performance. We evaluate the YOLOX-x object detector with pretrained COCO weights under different train-test-split configurations. Values are given in %.
(x_i, y_i, w_i, h_i), optionally with class c_i and confidence score s_i. As such, object detectors are an important part of most tracking pipelines, which extract features from detected objects to match them accordingly. For smart cities, 2D vehicle detection is important for vehicle localization.

Experimental Setup We use the YOLOX-x model [17] in the mmdetection [7] framework. Research on training detectors on synthetic data suggest that variety is more important than data size [13]. Therefore, for training and testing, we sample every tenth frame from all of our respective scenes and include it in the corresponding detection split.

Results Table 3 lists the results of our detection training. When using pretrained COCO weights only and thereby not training on Synthehicle at all, we obtain mediocre detection results. By visualizing false positives and negatives, we observe that YOLOX-x struggles to detect special CARLA vehicles, such as fire engines and cybertrucks. However, when fine-tuning on the Synthehicle train splits, performance increases significantly. As expected, performance on rain and night scenes is worse than on day and dawn scenes. Interestingly, we obtain the highest performance on dawn scenes. This might be because in dawn scenarios there are almost no shadows present in the scenes, potentially reducing the number of false positives. Compared to training and testing on weather-specific train and test splits, training on all splits and then testing on the individual test splits leads to increased performance. Thus, the variety within the training data is crucial for a well-generalizing model.

4.2. Vehicle Re-Identification

Object re-identification is an image retrieval problem in which a gallery set of m images G = \{I_{G,i}\}_{i=1}^{m}, I_{G,i} \in \mathbb{R}^{W \times H \times 3}, is ranked by similarity to a query image I_Q \in \mathbb{R}^{W \times H \times 3}, with the goal that the most similar gallery image belongs to the same identity (i.e., class) as the query image. In particular, images in object re-identification are obtained from multiple distinct cameras. The desired similarity measure in question is usually obtained by extracting features using a deep neural network and measuring their distances [28, 70, 46, 45, 25].

Table 4: Synthehicle Re-Identification Splits. We create five different splits for re-identification training.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training</th>
<th>Gallery</th>
<th>Query</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthehicle (day)</td>
<td>24 509</td>
<td>11 556</td>
<td>2 889</td>
<td>223</td>
</tr>
<tr>
<td>Synthehicle (dawn)</td>
<td>23 963</td>
<td>8 039</td>
<td>2 010</td>
<td>220</td>
</tr>
<tr>
<td>Synthehicle (rain)</td>
<td>22 171</td>
<td>7 036</td>
<td>1 760</td>
<td>205</td>
</tr>
<tr>
<td>Synthehicle (night)</td>
<td>22 943</td>
<td>8 775</td>
<td>2 194</td>
<td>205</td>
</tr>
<tr>
<td>Synthehicle (all)</td>
<td>93 586</td>
<td>35 407</td>
<td>8 852</td>
<td>853</td>
</tr>
</tbody>
</table>

Person and vehicle re-identification have received much attention in the past. Both tasks have to deal with similar problems like changing camera angles and lighting conditions. However, while people look and dress somewhat differently, vehicles can appear similarly. Additionally, vehicles provide almost no spatial-temporal pose information since, unlike humans, their shapes remain almost unchanged under movement. Important real datasets for vehicle re-identification are the VeRi dataset [35] with 776 vehicles captured over 20 cameras, and the AI City ReID dataset [56, 43], which is provided as part of the AICity-Challenge.

Experimental Setup We train vehicle re-identification on different train and test splits of Synthehicle using the fastreid [23] framework with a ResNet50 backbone [22] and instance normalization [47]. Train and test splits were created by sampling every tenth frame and cropping detections using ground truth annotations. Crops smaller than 50 × 50 were filtered out. The vehicle images are scaled to 256 × 256 pixels for the training and deformed if their bounding box is not quadratic. We always train up to 140 epochs with early stopping using the Adam optimizer [29]. During testing, we normalize the features in the Euclidean norm and use the cosine distance between feature vectors to calculate the cumulative matching characteristics (CMC) [42]. We report the CMC rank-1 accuracy (r1) and mean Average Precision (mAP) for evaluation. As in training, the images are scaled to 256 × 256 for testing.

Results Table 5 lists the results produced by the fastreid model with different train and test combinations. As expected, re-identification works best in the day and dawn scenes and performs slightly worse in the rain and night scenes. If trained on all scenes, performance is increased on all test splits. This is surprising since CARLA only provides a limited number of vehicle models, and many spawned objects will be identical in appearance, but different in vehicle ID. The results indicate that the “more data is better” assumption of deep learning also holds in this particular case.

4.3. Single-Camera Vehicle Tracking

In single-camera multi-object tracking-by-detection, the task is to generate consistent tracks from a time-ordered set of images I = \{I_t\}_t and corresponding object detections D = \{D_t\}_t, D_t = \{(x_i, y_i, w_i, h_i)\}_t (cf. Section 4.1), such that every distinct target is assigned an unambiguous and unique ID across all time frames. Occlusions, lighting variations, and false positive or negative detections make tracking challenging. Important trackers include, among others, DeepSORT [64], Tracktor [2], and CenterTrack [71].
Table 5: Vehicle Re-Identification Performance. The table shows a performance comparison under different train and test splits.

<table>
<thead>
<tr>
<th>Method</th>
<th>Train/Test</th>
<th>mAP (%)</th>
<th>rank 1 (%)</th>
<th>rank 5 (%)</th>
<th>rank 10 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All/All</td>
<td>47.82</td>
<td>51.55</td>
<td>72.29</td>
<td>80.03</td>
<td></td>
</tr>
<tr>
<td>Day/Day</td>
<td>59.89</td>
<td>62.02</td>
<td>77.12</td>
<td>85.04</td>
<td></td>
</tr>
<tr>
<td>Dawn/Dawn</td>
<td>47.57</td>
<td>51.17</td>
<td>72.30</td>
<td>80.47</td>
<td></td>
</tr>
<tr>
<td>Rain/Rain</td>
<td>39.08</td>
<td>48.44</td>
<td>72.31</td>
<td>80.29</td>
<td></td>
</tr>
<tr>
<td>Night/Night</td>
<td>27.04</td>
<td>36.84</td>
<td>58.76</td>
<td>67.41</td>
<td></td>
</tr>
<tr>
<td>All/Day</td>
<td>60.38</td>
<td>58.50</td>
<td>78.56</td>
<td>84.58</td>
<td></td>
</tr>
<tr>
<td>All/Dawn</td>
<td>53.21</td>
<td>55.75</td>
<td>82.51</td>
<td>84.87</td>
<td></td>
</tr>
<tr>
<td>All/Rain</td>
<td>45.33</td>
<td>52.60</td>
<td>75.66</td>
<td>83.76</td>
<td></td>
</tr>
<tr>
<td>All/Night</td>
<td>32.69</td>
<td>37.25</td>
<td>60.78</td>
<td>68.88</td>
<td></td>
</tr>
</tbody>
</table>

Experimental Setup We use the DeepSORT [64] tracker with the YOLOX-x models trained in Section 4.1 and the re-identification weights obtained from the experiments in Section 4.2. We follow the evaluation protocols of the CLEAR MOT metric, with multiple-object tracking accuracy and precision (MOTA and MOTP), and number of ground truth (GT), mostly tracked (MT), partially tracked (PT) and mostly lost (L) targets. We set the minimum detection height and width to 32 × 32 to filter small false positives and set the detection confidence threshold to 0.3. The DeepSORT parameter nn_budget, which decides how many appearance features are considered for a track, is set to 100.

Results Table 6 shows the results of DeepSORT on all test scenes. Note that results are averaged over the respective number of camera videos included in a scene for performance metrics. As expected, tracking in day and dawn scenes generally yields superior performance than tracking on rain and night scenes. However, for Town06-O-night, DeepSORT performs better than in other scenes with the same camera setup. We conjecture that the 10 fps frame rate is sufficiently large to track objects in a scene with comparatively low density (cf. Table 2). Note also that scenes with different ambience configurations are rendered with different views and number of vehicles. Only the camera setup is fixed. Overall, DeepSORT-based single-camera tracking on Synthethicicle generates satisfying baseline results.

4.4. Multi-Vehicle Multi-Camera Tracking

MTMCT is the extension of single-camera tracking to a multiple-camera setup, where the input set of images now comes from K distinct views \( I = (I^{(c_1)}_1, \ldots, I^{(c_K)}_K) \), and the detections are obtained accordingly as \( D^{(c_i)} = \{D^{(c_i)}_{ij}\}_{i=1, \ldots, K} \). Generally, there is no restriction on the camera topology and field of views can be overlapping or non-overlapping. In practice, many MTMC trackers [52, 48] first apply a single-camera tracker on the individual camera videos to obtain single-camera results (i.e., tracklets) \( T_1, \ldots, T_K \), and then match those sets of local tracklets into a set of global tracklets \( T \).

Method We use a method based on ELECTRICITY [48], a 2020 challenge winner for the CityFlow dataset, with the single-camera trackers \( T_1, \ldots, T_K \) obtained by applying DeepSORT as in Section 4.3. For each tracklet, a query image is chosen, and all other images of that tracklet are selected to be gallery images. Using the feature extractor trained in Section 4.2, a query feature matrix \( Q \in \mathbb{R}^{n \times f} \) and a gallery feature matrix \( G \in \mathbb{R}^{m \times f} \) are built, where \( n \) is the number of tracklets, \( m \) is the number of gallery images, and \( f \) is the feature embedding size. After normalizing all query and probe features, i.e., \( \|q_{ij}\|_2 = 1 \forall i = 1, \ldots, n \) and \( \|q_j\|_2 = 1 \forall j = 1, \ldots, m, m \) obtain the cosine distance matrix \( D \) as

\[
D = QG^T \in \mathbb{R}^{n \times m}.
\]

The multi-camera tracking result is then derived from \( D \) by merging the tracklets \( T_i \) and \( T_j \) if and only if both the feature distance between the query image of \( T_i \) and the gallery images of \( T_j \), and the feature distance between the query image of \( T_j \) and the gallery images of \( T_i \) are below a specified threshold \( \theta \), i.e.,

\[
D_{ij} < \theta \quad \text{and} \quad D_{ji} < \theta.
\]

Experimental Setup We apply the method described above with \( \theta = 0.8 \) on all test scenes. For evaluation, we report the IDF1, IDP and IDR metrics for multi-camera tracking as suggested by [50]. Features are extracted on single-camera tracklets obtained in Section 4.3 using the re-identification network trained in Section 4.2.

Results Table 6 list the performance of the multi-camera tracker on Synthethicicle. Since the tracker is based on pre-extracted single-camera tracklets, there is a correlation between single-camera performance and multi-camera performance. The multi-camera tracker relies solely on re-identification and performance sometimes decreases in night scenes (e.g., in Town10HD-O-night). Figure 5 shows the influence of scene density and single-camera IDF1 performance on multi-camera IDF1 performance. Dense scenes increase the difficulty for our multi-camera tracking, since more gallery and query features have to be considered during matching, leading to potential association errors.

5. Conclusion

We have presented a massive synthetic dataset for tracking vehicles across multiple overlapping and non-
Table 6: Single-Camera and Multi-Camera Vehicle Tracking Performance. Performance of single- and multi-camera tracking on Synthecihcle. Single-camera performance results are obtained for every individual camera in the scene, and then averaged for this table, while the discrete statistics (GT, ML, PT, ML, IDs) are aggregated by summation.

![Image of a table showing performance metrics for different scenes and conditions.](image)

**Figure 5: Results analysis.** The scene density has a strong influence on multi-camera tracking performance (a). The quality of single-camera tracklets influences the multi-camera tracking performance, but less than expected (b).

overlapping cameras in various scenes with a wide variety of data annotations previously not included in similar datasets, such as semantic, instance and panoptic segmentations, depth maps, and over 4 million annotated 2D and 3D bounding boxes. With 17 hours of video material and 340 cameras, it is the largest available MTMCT dataset. The ambiance configurations included in our dataset allow for exploring multi-vehicle tracking under challenging conditions. We have demonstrated the performance of different baselines for vehicle detection, re-identification, and single- and multi-camera tracking tasks. Results on these tasks indicate that Synthecihcle is a complex dataset with diverse and challenging scenarios. While the focus of our analysis has been MTMCT, our annotations can potentially enable the exploration of new tasks, such as 3D multi-target multi-camera tracking or multi-camera tracking and segmentation. Recording and labeling real datasets is a time-consuming task. Furthermore, in the case of large datasets, it cannot be assumed that persons contained in the dataset have given their consent. One-sided datasets could also reinforce problematic biases. Synthetic data, on the other hand, can be collected without any privacy violations and biases could potentially also be compensated for in such data.

**Acknowledgement** We gratefully acknowledge the financial support from Deutsche Forschungsgemeinschaft (DFG) under grant number RI 658/25-2.
References


