Journal of the Brazilian Computer Society, 2025, 31:1, doi: 10.5753/jbcs.2025.5869

ENDLESS: An End-to-End Framework for Urban Synthetic Dataset Generation

Amadeo Tato Cota Neto [**Voxar Labs, Centro de Informática, UFPE** | ***atcn@cin.ufpe.br***]
Willams de Lima Costa [**Voxar Labs, Centro de Informática, UFPE** | ***wlc2@cin.ufpe.br***]
Veronica Teichrieb [**Voxar Labs, Centro de Informática, UFPE** | ***vt@cin.ufpe.br***]
João Marcelo X. N. Teixeira [**EACH - USP, Voxar Labs, UFPE** | ***jmxnt@cin.ufpe.br***]

Universidade Federal de Pernambuco, Av. Prof. Moraes Rego, 1235 Cidade Universitária, Recife, PE, 50670-901, Brazil.

Received: 09 April 2025 • **Accepted:** 24 June 2025 • **Published:** 29 September 2025

Abstract Computer vision models are fundamental for smart city applications. These models enable the city to interpret visual data, obtained from sensors such as surveillance cameras, to optimize its tasks and positively impact the citizens' lives. However, these models require ever-growing amounts of labeled data for training, which is expensive and raises ethical concerns when collected in the real world. Conversely, 3D engines and simulators allow the cost-effective and large-scale generation of automatically annotated synthetic data. This work proposes a synthetic dataset generator for the smart cities field using the CARLA simulator. The proposed generator allows the end-to-end generation of massive datasets with a single command, which includes the simulation of city assets, such as vehicles and pedestrians, and the recording and annotation of visual data. To demonstrate the generator's effectiveness, a dataset with over 300K annotated frames was generated and compared with other state-of-the-art datasets. The comparison results show that the proposed generator is capable of producing datasets comparable to the state of the art in terms of data volume and number of annotations. It's expected that the proposed generator could be used to create useful datasets for training and evaluating computer vision models in the smart cities area. It's also expected that this work brings attention to the synthetic data usage for smart city models.

Keywords: Synthetic Data, Smart Cities, Computer Vision

1 Introduction

Urban spaces must be capable of ensuring the well-being of their citizens, guaranteeing them a pleasant quality of life and easy access to resources and services. However, the rapid urbanization of the world makes cities increasingly populated, bringing challenges to the management of resources and the assurance of a enhanced quality of life for their citizens.

In light of these problems, smart cities integrate advanced technological solutions to improve the quality of life of their residents and to optimize their tasks. Technologies from the fields of AI, IoT, and big data are commonly used in smart city applications to achieve this, allowing the city to understand its context and make informed decisions about it [Silva *et al.*, 2018; Zaman *et al.*, 2024].

In this context, CV models play a fundamental role in enabling smart cities to analyze and understand events occurring in their streets by processing visual data collected from sensors such as RGB cameras [Syahidi *et al.*, 2023]. These models can be applied in diverse contexts, such as traffic monitoring [Barthélemy *et al.*, 2019], accident detection [Adewopo *et al.*, 2023], and safety [Yar *et al.*, 2023].

However, supervised deep learning models require an increasing amount of labeled data to be trained [Dosovitskiy *et al.*, 2020]. Traditionally, this data is obtained by collecting images and videos from the real world for a given period and annotating them afterward [Geiger *et al.*, 2012; Cordts *et al.*, 2016]. However, this approach is time-consuming (requiring more than 1.5h per image for fine annotations in Cordts *et al.* [2016]), expensive, and raises concerns regarding the privacy of the data and possible biases associated with the dataset.

On the other hand, synthetic datasets are collections of data generated in digital environments, such as game engines [Gaidon *et al.*, 2016], specific software [Herzog *et al.*, 2023], or even computer games [Richter *et al.*, 2016]. These datasets simulate real-world physics, environments, and rules in the digital world, enabling the creation of data that closely resembles real-world datasets [Gaidon *et al.*, 2016].

Compared to real-world datasets, synthetic datasets are a faster and more cost-effective alternative to gathering data to train CV models. These datasets can be generated with a massive amount of data, are automatically annotated, and offer better control of the dataset environment, allowing adjustments on the data diversity [Pathiraja *et al.*, 2024] and allowing the recording of rare real-world events [Gaidon *et al.*, 2016].

However, since the real world cannot be fully simulated in a virtual environment, a reality gap [Tobin *et al.*, 2017] may be present in synthetic datasets. This gap may arise from the discrepancies between simulation visuals and physics in the real world, possibly leading to implausible interactions or unnatural events. Additionally, it is necessary to add diverse scenarios on synthetic datasets to prevent bias. Despite these challenges, prior works have shown that the reality gap can be mitigated by adjusting dataset features or using specialized training techniques [Fabbri *et al.*, 2021; Tobin *et al.*, 2017; Gaidon *et al.*, 2016; Ros *et al.*, 2016].

For instance, Gaidon *et al.* [2016] and Ros *et al.* [2016] demonstrated that training models with a combination of synthetic and real data can lead to improved performance compared to training on real data only. Specifically, Gaidon *et al.* [2016] enhanced the performance of multi-object tracking models by pre-training on synthetic data and fine-tuning them on real data. Similarly, Ros *et al.* [2016] mixed real and synthetic data during the training of semantic segmentation models to obtain improved results. On the other hand, Tobin *et al.* [2017] and Fabbri *et al.* [2021] showed that it is possible to achieve state-of-the-art results using only synthetic data. This was achieved by using Domain Randomization in object detection tasks for robotics in Tobin *et al.* [2017] and by increasing the data diversity in Fabbri *et al.* [2021] for pedestrian detection, tracking, and ReID.

However, while there are plenty of urban synthetic datasets available, they often are created from very specific viewpoints. For example, synthetic datasets for autonomous driving applications are often generated using cameras placed on top of vehicles or in the perspective of vehicles' hoods [Ros *et al.*, 2016; Gaidon *et al.*, 2016]. Other datasets contain top-down camera views that are useful for drone tasks [Turkcan *et al.*, 2024], for example. Although useful, these datasets are not focused on the city as a whole, which is limiting to tasks in which the city should be an active agent that perceives its context and actuates on it.

Additionally, even though these synthetic datasets can be modified, their frameworks often do not provide an end-to-end approach, which makes the generation of new data difficult. We consider a dataset generation framework to be end-to-end if it encapsulates all steps of dataset generation in a single command.

In this work, we propose the ENDLESS framework, an end-to-end dataset generation framework developed on top of the CARLA Simulator [Dosovitskiy *et al.*, 2017]. The framework enables the generation of large urban synthetic datasets by executing a single script after the initial CARLA setup. The generated datasets are automatically annotated with 2D bounding-boxes, instance segmentation, and depth maps. In addition, the framework allows for changes in generation parameters (such as weather, city maps, and others), enabling users to generate customized datasets.

To demonstrate the effectiveness of our framework, we used it to generate a novel dataset with more than 300,000 frames and compared it with state-of-the-art urban synthetic datasets. Our analysis shows that our framework can generate datasets with a number of frames and annotations comparable to the state-of-the-art.

This work is structured as follows: first, in section 2, we provide the necessary background to support this work. Next, in section 3, we present related works of urban synthetic datasets and dataset generation tools. After that, we present our framework at section 4. Then, we introduce our generated dataset and analyze it at section 5. Furthermore, we compare our dataset to others from the state-of-the-art urban synthetic datasets in section 6. Finally, we present our conclusions and discuss limitations and future works in section 7.

2 Background

The constant advances in the field of Deep Learning led to the widespread adoption of these models in various tasks, making them present in multiple aspects of everyday life. In particular, Deep Learning models for CV can interpret the real world through images and videos, making them essential for smart city tasks [Syahidi *et al.*, 2023].

These models typically use a supervised learning approach, in which the model learns from examples provided in a labeled dataset. To make this training possible, the dataset must be annotated according to the task the model is designed to perform. However, as the field advances, Deep Learning models are increasingly large and require ever-growing amounts of data to be trained [Dosovitskiy *et al.*, 2020]. Nonetheless, obtaining fine annotations of real-world datasets is a time-consuming task [Cordts *et al.*, 2016] and the recorded dataset may contain intrinsic biases related to the location and conditions in which it was recorded.

In this section, we provide the necessary background to support our work and to better understand its placement within the current literature. First, we provide an overview of the fields of smart cities and CV. Second, we present the types of annotations used in CV models. Finally, we present simulation engines used to generate synthetic datasets.

2.1 Smart Cities

In an increasingly urbanized world, cities must be able to address challenges related to citizens' quality of life, service provision, air pollution, and other challenges arising from constant urbanization. The smart city concept emerges to describe a city that can tackle these challenges and thrive.

Although in increasing popularity, a unique definition of what constitutes a smart city is not universally accepted and many works provide different definitions of it [Chourabi *et al.*, 2012]. Giffinger *et al.* [2007] provide a holistic definition of the term, involving various sectors of urban areas. According to them, a smart city should thrive in economy, people, governance, mobility, environment, and living; and their citizens must have active participation in the city. To better describe these characteristics, the authors provided 33 factors that provide a better understanding of them.

A concept also associated with smart cities is that these cities should be able to capture information about their context and interpret it to enable informed decision-making [Zaman *et al.*, 2024; Silva *et al.*, 2018]. In this regard, ICT solutions, such as AI, big data, and IoT, are widely explored in research on the field [Silva *et al.*, 2018; Zaman *et al.*, 2024].

IoT devices enable information exchange between city assets through the internet, creating a network where each asset contributes to improving its own performance and the city as a whole. These devices also provide real-time data from multiple regions and sectors of the city, providing a holistic overview of its context [Zaman *et al.*, 2024]. The high volume of data collected from all these devices can be processed by big data techniques, enabling the filtering of redundant or erroneous data and normalizing it. The processed data can then be stored for future use and be analyzed to generate insights on how to improve different aspects of the city [Silva *et al.*, 2018]. The raw or processed obtained device data can be used by AI models to enable the city to detect events and autonomously react to them through its devices [Adewopo *et al.*, 2023; Barthélemy *et al.*, 2019].

To illustrate how these technologies improve urban dynamics in smart cities, imagine the following situation: a traffic accident occurred on a busy avenue just before rush hour. A smart city should be able to automatically detect the accident, through surveillance cameras for example, and notify emergency services to assist those involved. At the same time, the city would work to prevent traffic jams by informing citizens of the incident and suggesting alternative routes to them. In a V2X scenario, the city would also be able to communicate directly with autonomous vehicles so that they could recalculate their routes.

2.2 Computer Vision

Computer vision is a field of AI that studies how computers can interpret visual inputs, such as images and videos, in an attempt to replicate the mechanisms of the human brain and vision. This field is rapidly expanding due to improvements on Deep Learning models, which enabled various CV tasks to be performed.

Classic algorithms in the field use image processing and statistics to extract features from images. These features could be used as input for traditional machine learning models to perform CV tasks. Today, Deep Learning models have become the new standard in the field, allowing automatic extraction of features from images and more accurate predictions.

Most Deep Learning models for CV are trained using supervised learning, meaning they generally require labeled data for training. Just as other AI models, labeled datasets for CV contain input data and ground-truth annotations for those inputs [Geiger *et al.*, 2012].

Image classification may be the best-known task in the field. In this task, the model's objective is to predict the class of an image from a predefined set of classes. The LeNet model [LeCun *et al.*, 1998] enabled handwritten digit recognition by classifying grayscale images of digits as one of ten possible classes. Today, with the advancements in the Deep Learning field, image classification models are capable of identifying the class of an image among thousands of categories for which the model has been trained [Dosovitskiy *et al.*, 2020].

Object detection models also provide the class of objects from an image, but they also provide the location and dimension of these objects. The location of the objects is given by enclosing the object in bounding-boxes [Redmon *et al.*, 2016], which will be better explained in Section 2.3.1.

Segmentation models take a step further by classifying each pixel in the image rather than just detecting objects, providing more detailed results [He *et al.*, 2017]. These models require segmentation maps for training, which will be detailed further.

Depth estimation is another common task in CV, enabling the generation of depth maps, which estimate the distance of each pixel in an image from the camera [Bhat *et al.*, 2023]. More details on depth maps can be found further.

2.3 Common Computer Vision Annotations

As mentioned before, the training of CV models usually requires annotated data. The annotation process is usually slow and requires human annotators to manually obtain ground-truth data from images or the usage of specific hardware or software solutions. Nowadays, tools such as Roboflow[1] and CVAT[2] speed up the annotation processes by using AI techniques, but still requires human annotators to generate finer annotations.

In this section, we present common annotations used to train CV computer vision models. We specifically focus on bounding-boxes, segmentation, and depth maps once these are the annotations we provide in our synthetic data generation framework.

2.3.1 Bounding-Boxes

Bounding-Boxes are a type of annotation used to specify the position and dimensions of objects in images. This is done by enclosing the object within a box that contains it, as shown in Figure 1. This annotation is fundamental for training AI models in tasks that require the location or dimension of objects in the image, such as object detection [Redmon *et al.*, 2016].

Just as there are various ways to represent squares, there are various ways to represent bounding-boxes of objects. The YOLO models [Redmon *et al.*, 2016] utilize bounding-boxes represented by the center of the box and its dimensions. Other works represent the bounding-boxes storing their top-left and bottom-right vertices coordinates such as the Faster-RCNN [Ren *et al.*, 2015].

Extending the conventional 2D bounding-boxes, 3D bounding-boxes contain additional depth information, using rectangles to enclose objects. The additional depth information makes 3D bounding-boxes useful for applications in the field of autonomous driving, 3D reconstruction, and extended realities. However, obtaining 3D bounding-boxes is more challenging as it requires depth data and a more complex annotation process.

2.3.2 Segmentation Maps

Segmentation maps are pixel-level annotations that encode information in the RGB values of each pixel of an image, making it rich in detail and precision. There are different types of segmentation maps, which differ in the information stored in the RGB values. For semantic segmentation maps, the RGB value stores the class of the object comprising the pixel. For instance, segmentation maps, both the object class and a unique ID for each object are encoded in the RGB values in the image. This is often achieved by storing the object class in one channel and the object ID in the remaining channels. Figure 2 provides examples of instance segmentation maps.

[1]Available at: https://roboflow.com/. Accessed on: Mar. 18, 2025
[2]Available at: https://www.cvat.ai/. Accessed on: Mar. 18, 2025

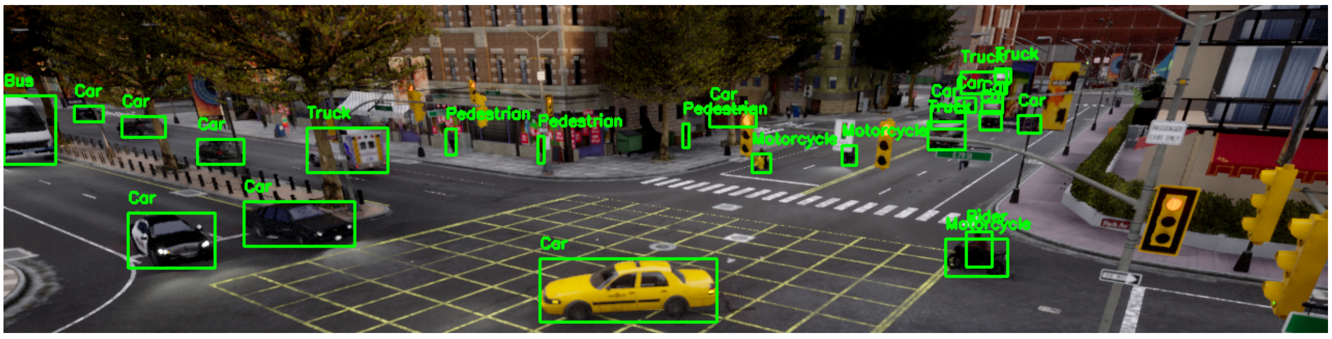


Figure 1. Example of 2D bounding boxes.

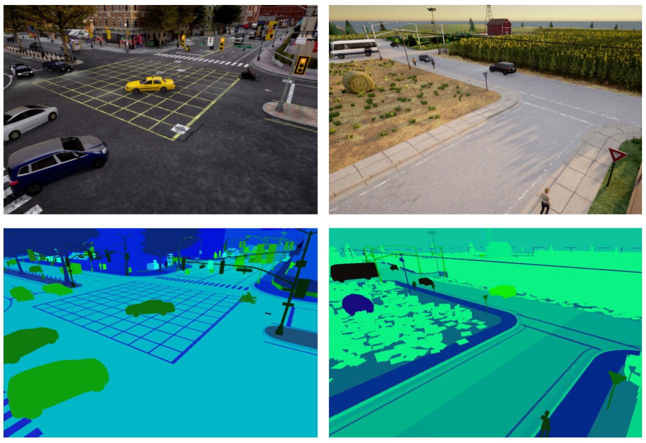


Figure 2. Example of instance segmentation map.

2.3.3 Depth Maps

Depth maps store the distance from a given pixel to the camera, typically in meters, adding depth information to 2D images (as illustrated in Figure 3). Obtaining these maps required expensive sensors such as LiDAR, stereo cameras, or structured light systems, limiting the obtaining of this data.

Advancements in Deep Learning and Computer Vision techniques have made depth map generation more accessible by enabling precise depth estimation. Models like ZoeDepth [Bhat *et al.*, 2023] can produce high-quality depth maps from monocular images captured using standard RGB cameras, significantly lowering the costs associated with specialized sensors.

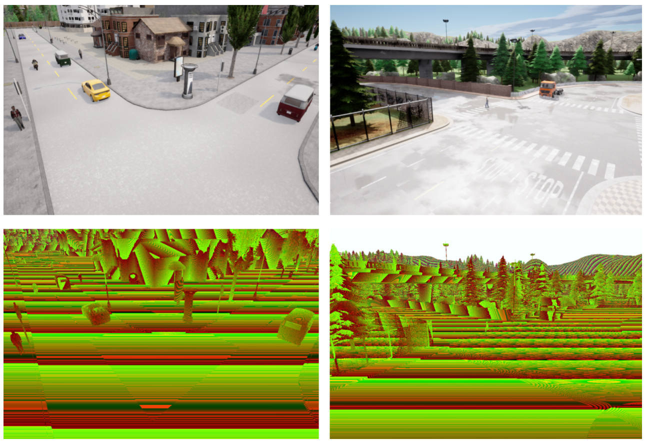


Figure 3. Example of depth maps.

2.4 Simulation engines for synthetic datasets

Advancements in computer graphics and game development fields resulted in improved realism in digital environments, both in terms of graphical quality and physical accuracy. In this context, simulation engines emerge as computer graphics software that enables the simulation of real-world rules and environments in a digital setting, allowing experiments to be conducted in the virtual world. In particular, the fields of autonomous vehicles and smart cities have strong incentives to use these tools, as simulations make it possible to carry out experiments that would otherwise be costly or difficult to carry out in the real world [Dosovitskiy *et al.*, 2017].

Various of these engines make it possible to obtain visual data through simulated sensors, which include RGB images, depth and segmentation maps, LiDAR scans, and others. The availability of these visual sensors combined with the realism of these engines enables their use to generate synthetic datasets that accurately replicate real-world dynamics, making them suitable for training CV models.

The BeamNG.tech[3] is a simulation engine aimed for applications on the autonomous vehicles and driver training fields. The engine has its own physical simulator, which uses a custom soft-body physics that helps to realistically simulate vehicle kinematics.

The engine provides a set of sensors for data acquisition through simulation, including cameras, LiDAR, IMU, and ultrasonic sensors. Additionally, it enables the acquisition of ground-truth data, such as bounding-boxes and segmentation. These features enable synthetic datasets to be recorded using the engine, exploiting the realistic vehicle kinematics.

One of the most notable simulation engines is the CARLA simulator [Dosovitskiy *et al.*, 2017], built using a fork of the popular Unreal Engine[4] and focused on experiments with autonomous driving models. The simulator provides digital cities in which vehicles can roam and simulate traffic agents such as vehicles, pedestrians, traffic lights, and others. Figure 4 shows the CARLA simulation interface on Unreal Engine 4.

The simulator also contains a great sensor suite that was originally designed to simulate how an autonomous car could interact with the environment. These sensors can be either attached to vehicles or placed around the available cities and include RGB, depth, and segmentation cameras and others

[3]Available at: https://beamng.tech/. Accessed on: Mar. 17, 2025.
[4]Available at: https://www.unrealengine.com/. Accessed on: Mar. 17, 2025.

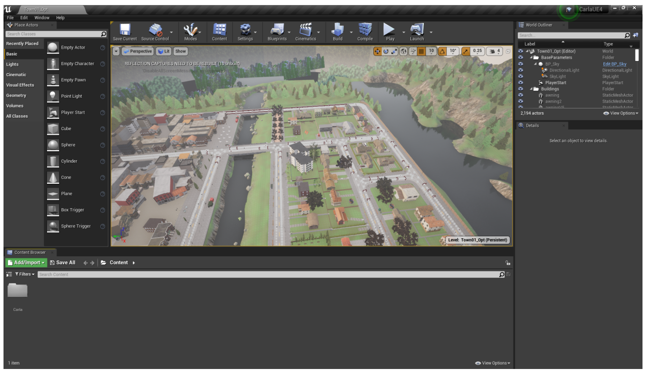


Figure 4. Carla simulator interface on Unreal Engine 4.

such as LiDAR, IMU, and optical flow.

Thanks to the traffic realism, great range of sensors, and open-source nature, the CARLA simulator is extensively used to record synthetic data aimed for urban tasks [Herzog *et al*., 2023; Kloukiniotis *et al*., 2022; Deschaud, 2021; Deschaud *et al*., 2021].

3 Related Work

In this section, we present related works in the field of urban synthetic datasets and synthetic data generation tools. We describe an overview of the field and prominent works on it. Subsequently, we compare these works with our own to discuss our contributions to the current literature.

3.1 Urban Synthetic Datasets

Urban synthetic datasets simulate elements from city streets and avenues, such as vehicles, pedestrians, and traffic lights, which make them especially suitable for autonomous vehicles and smart city tasks. Due to the versatility of synthetic data, there are synthetic datasets available for diverse tasks, such as vehicle tracking [Gaidon *et al*., 2016; Herzog *et al*., 2023], pedestrian detection [Stauner *et al*., 2022; Fabbri *et al*., 2021], and 3D mapping [Deschaud *et al*., 2021].

The camera positioning on urban synthetic datasets can also be tailored for different application domains. Examples of camera positioning found on available urban synthetic datasets are egocentric vehicle cameras for autonomous vehicles tasks [Gaidon *et al*., 2016; Ros *et al*., 2016], aerial cameras [Turkcan *et al*., 2024] and surveillance cameras [Herzog *et al*., 2023].

The Synthehicle dataset [Herzog *et al*., 2023] is a synthetic dataset built using the CARLA simulator [Dosovitskiy *et al*., 2017] and contains 17 hours of video recorded by cameras positioned similarly to surveillance cameras. It contains annotations for 2D and 3D bounding-boxes, depth, multi-camera tracking, and segmentation (instance, class, and panoptic).

A medium-altitude aerial dataset was developed to demonstrate the effectivity of the "Boundless" simulator, developed by Turkcan *et al*. [2024]. The dataset consists of 8K frames collected over a single intersection with 2D and 3D bounding-box annotations. The authors further created similar datasets using CARLA's simulator (22K frames) and their simulator to create a digital twin of a real-world intersection (8.7K frames).

We noticed that existing works predominantly focus on autonomous vehicles tasks, containing egocentric camera viewpoints that capture the vehicle perspective [Deschaud, 2021; Gaidon *et al*., 2016; Ros *et al*., 2016; Kloukiniotis *et al*., 2022] while overlooking surveillance camera perspectives, which are crucial for smart city applications. Furthermore, many of these datasets do not offer a simple end-to-end process for expanding them with new data under customizable settings, keeping their size fixed.

3.2 Tools for synthetic data generation

Synthetic datasets can be generated by various tools, including game engines, simulation engines, and even computer games [Paulin and Ivasic-Kos, 2023]. Game engines, such as Unity[5] and Unreal Engine[6], are commonly employed to generate synthetic data from simulated environments, which can be done by obtaining data from simulated virtual cities [Li *et al*., 2023; Ros *et al*., 2016; Kerim *et al*., 2021] or by digitally cloning real-world dataset scenes [Gaidon *et al*., 2016]. The increasing realism of computer games has also made them a valuable source for data generation. A notable example is Grand Theft Auto V[7], developed by Rockstar Games, which was used in the works of Fabbri *et al*. [2021] and Richter *et al*. [2016] to generate synthetic datasets.

Simulation engines, such as BeamNG.tech[8] and CARLA simulator Dosovitskiy *et al*. [2017]. are also valuable tools for generating synthetic datasets. More details about simulation engines, including CARLA and BeamNG.tech, can be found at subsection 2.4.

Boundless [Turkcan *et al*., 2024] is a dataset generation tool built on top of the CitySample asset[9] from Unreal Engine. It can generate photorealistic RGB images with associated 2D and 3D bounding-box annotations. The simulator supports dynamic weather conditions and time-of-day progression throughout the same simulation.

The NOVA [Kerim *et al*., 2021] framework is also focused on generating synthetic data from urban scenes but with particular emphasis on virtual humans. The generator allows various camera positions, including surveillance positioning, and features four scenarios in its gallery. A procedural human generator system randomly selects features from a predefined set to create diverse, highly varied virtual humans. Ground-truth annotation is generated for 2D bounding-boxes (for humans), body part segmentation, body pose, optical flow, depth, surface normals, and instance and semantic segmentation.

Current methods for generating synthetic datasets still require some level of user intervention during the generation process. This intervention can range from developing a data collection system to making smaller adjustments, such as modifying weather, map settings, or camera positioning.

[5]Available at: https://unity.com/. Accessed on: Mar. 17, 2025.
[6]Available at: https://www.unrealengine.com/. Accessed on: Mar. 17, 2025.
[7]Available at: https://www.rockstargames.com/gta-v. Accessed on: Mar. 17, 2025.
[8]Available at: https://beamng.tech/. Accessed on: Mar. 17, 2025.
[9]Available at: https://www.fab.com/listings/4898e707-7855-404b-af0e-a505ee690e68. Accessed on: Mar. 17, 2025.

Comparison with related works In this work, we propose an end-to-end synthetic dataset generation framework capable of producing large and diverse datasets with a single command. The datasets are easily expandable, as the generator can continuously produce new data. Furthermore, the generated data is captured in the perspectives of surveillance cameras, making the generated datasets well suited for training smart city models.

Comparing our work with the aforementioned related works, we can map how our proposal differs in two ways:

1. Our framework for synthetic dataset generation can create customized and on-demand datasets in an end-to-end fashion, allowing the generation of large datasets and eliminating the need to manually define how the data will be collected.
2. Different from most works, our generated synthetic dataset was designed for smart city applications and is easily expandable due to our framework.

4 The ENDLESS dataset generation framework

In this section, we describe how the ENDLESS framework was implemented and provide an overview of the data generation pipeline. We start the section with a brief overview of our framework. After that, we present the reasons we developed ENDLESS on top of the CARLA Simulator [Dosovitskiy *et al*., 2017]. Finally, we provide an overview of the data generation pipeline from the user interaction to the post-processing step, in which the videos are generated from the recorded data.

4.1 Overview of the Framework

The ENDLESS framework enables the end-to-end generation of synthetic datasets for CV tasks in the context of smart cities. The data includes elements of urban streets, such as vehicles and pedestrians, and is captured from the simulated perspective of surveillance cameras. Annotations for the generated data are automatically produced with respect to instance segmentation, depth, and 2D bounding boxes.

Our framework can generate diverse data that reflect various real-world scenarios. By leveraging the CARLA Simulator [Dosovitskiy *et al*., 2017] features, our framework can capture data from varied urban settings in distinct environmental conditions and over different traffic and crowd settings. Users can customize the data generated by our framework adjusting parameters such as crowd density, number of vehicles, enabled city maps, weather conditions, and time of day.

Once generated, datasets generated by our framework are easily expandable thanks to its end-to-end pipeline. By executing a single Python script, users can generate new data to increase their current dataset size or enhance the representation of specific scenarios.

4.2 Dataset Generator

In order to allow a generator to simulate dynamics from cities and capture relevant ground-truth data, we argue that there are several benefits in using established simulators from the fields of autonomous driving and smart cities. The main benefit is the access to built-in tools that help to simulate city dynamics automatically, such as traffic and pedestrian flow, and to capture data from the environment. Without needing to develop these features from scratch, researchers can focus on generating meaningful and comprehensive data, which may lead to higher-quality datasets.

Our main reasons for building our simulation on top of CARLA were its large catalog of assets, widespread adoption in the literature, and the vast range of built-in features. These features include automatic control of city assets, weather manipulation, and sensor simulation for data retrieval. More details about the CARLA simulator can be found at subsection 2.4.

4.3 Data Generation Pipeline

Our framework follows an automated end-to-end pipeline, simplifying the dataset generation process to the execution of a single Python script after the initial CARLA setup. We represent the full generation pipeline in Figure 5 and explain the steps from the user execution to dataset generation in the subsections below.

4.3.1 User Interaction

As we built our framework on top of the CARLA simulator, users must have it installed on their machines to use our framework. After starting the simulator, our framework can be executed by running a single Python script using the operational system terminal.

To allow users to create customized datasets easily, we provided arguments for the Python script. These arguments allow users to adjust various settings in their datasets. Specifically, users can define the target number of vehicles and pedestrians, enabling the generation of data from environments with varying levels of crowd density—from highly crowded cities to areas with fewer agents. Additionally, users can toggle settings for weather conditions, times of day, and city maps, as outlined in Table 1.

Table 1. Possible values for map, weather, and time of day

Setting	Possible Values
Maps	Town01_Opt, Town02_Opt, Town03, Town04_Opt, Town05_Opt, Town06_Opt, Town07_Opt, Town10HD_Opt
Weathers	clear_sky, rainy, cloudy, foggy, after_rain
Times of Day	morning, afternoon, night

Furthermore, these arguments offer control over technical settings related to the generated data, such as frame resolution, video frame rate, and video duration. Users can also specify the number of videos to be generated, allowing for further customization of the data.

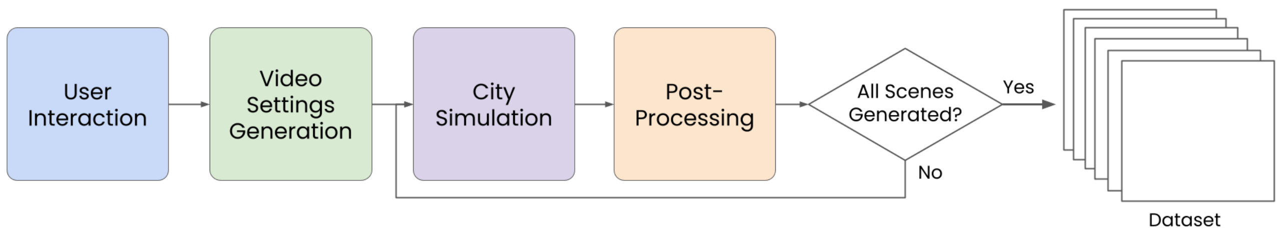


Figure 5. Data generation pipeline

4.3.2 Video Settings Generation

Once executed, our framework receives the user-defined settings and begins generating the dataset by defining the scene settings. We define a scene as the simulated environment from where data will be collected in one iteration of the data generation process. The scene settings include the desired number of pedestrians and vehicles, the name of the city that will be loaded on the CARLA simulator, the weather and time of day that will be simulated, and a pair of camera positions.

Using the user-defined settings, our framework obtains all possible scene settings by calculating the cartesian product from the values of city, weather, time of day, and distinct camera pairs. The resulting combination list is shuffled to prevent scenes with similar settings from being generated sequentially.

Since the number of possible combinations can differ from the desired number of videos, the system does one of the following changes:

- If the number of combinations is **higher** than the desired number of videos, the system selects a subset of combinations to match the number of videos;
- If the number of combinations is **lower** than the required number of videos, the system oversamples the combination list by repeating elements proportionally to match the required count.

Finally, the system groups combinations by city to minimize the number of times the CARLA simulator needs to change the active city. The settings from each scene are exported to JSON files, becoming available after the generation as metadata.

4.3.3 City Simulation and Camera Positioning

After the scene settings are generated, the framework retrieves the first of them to obtain the settings for the city simulation process. The city simulation process follows the steps of Algorithm 1.

Firstly, our framework loads the city map from the current scene. To allow the generation of diverse environments, we utilized the eight standard city maps from the CARLA catalog, which range from downtown streets to rural cornfields.

After that, we simulate the time of day and weather specified on the scene settings. By using the CARLA Weather API, we predefined a set of weather and time-of-day conditions that can be applied in each scene. The available weather conditions include clear sky, rain, clouds, fog, and post-rain; while the possible time of day includes morning, afternoon, and night. To ensure that the generated data can reflect diverse and challenging real-world scenarios, we designed our framework to enable the weather and time-of-day conditions to be adjusted independently. Figure 6 presents the weather and time-of-day conditions available in the simulation.

The city is then populated with agents to simulate the city dynamics. To do this, pedestrians and vehicles are randomly selected from the CARLA catalog and spawned in predefined positions on the city map. Each agent moves to random positions in a non-oriented and infinite path. In particular, vehicles are controlled by the CARLA's TrafficManager, which is responsible for their paths and behaviors, such as stopping at red traffic lights or stop signs, slowing down in speed signs, turning on the vehicle lights in reaction to traffic events, and so on.

To enable the data recording, the pair of cameras is positioned in the city according to the scene settings. To grant diversity of views on the recorded data, we positioned 5 cameras for each city simulating positions of real-world surveillance cameras, similar to Herzog *et al.* [2023]. The cameras were spread around the city attached to building walls, streetlights, fences, and other structures. During the camera positioning, we tried to maximize the diversity of scenarios that the camera could capture. We included cameras capturing data from road intersections, highways, sloped roads, rural roads, urban parks, and so on.

4.3.4 Recording Data

While the city is being simulated, the framework uses the pairs of cameras, previously positioned, to record RGB images, depth maps, and instance segmentation maps from the environment for the video duration. We can achieve this by placing RGB, depth, and instance segmentation cameras in the same location and with the same settings. This allows us to obtain a perfect match between the sensors' data without any calibration process.

The data from the cameras are recorded as PNG images and named according to the current simulation frame. Instance segmentation images are stored in a way that the red channel defines the pixel class and the green and blue channels, the unique ID from the pixel's object. Depth is stored in RGB images in a way that the distance in meters from the

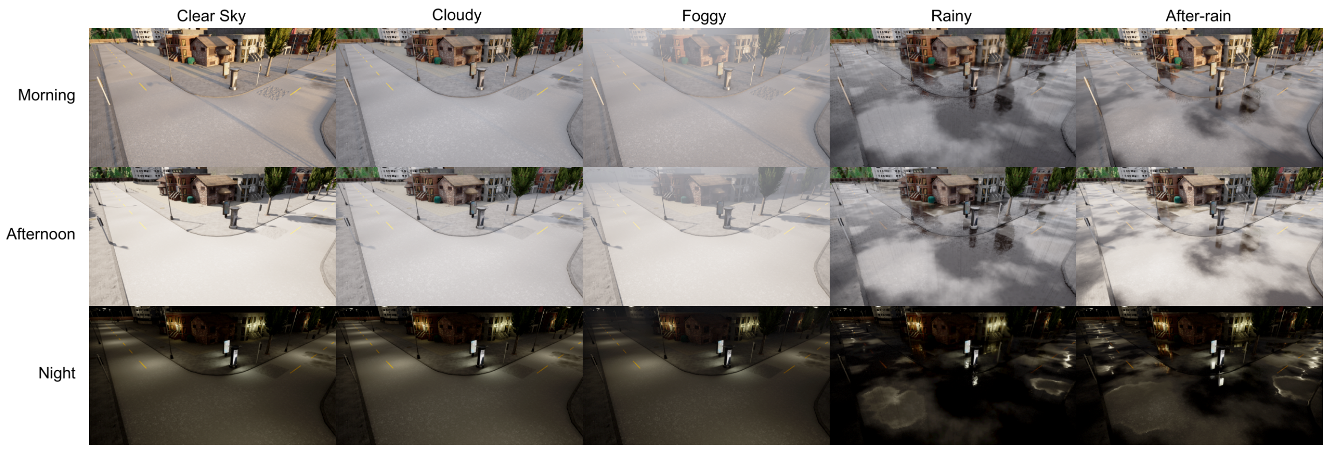


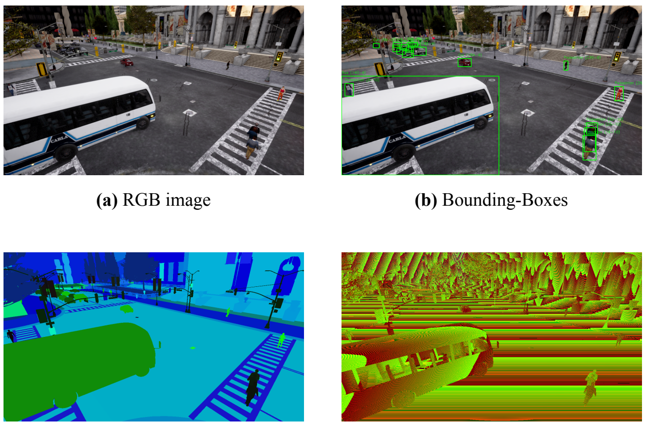
Figure 6. Weather and time-of-day conditions

object to the camera can be calculated, as described in the CARLA documentation[10], with the following equation:

$$D_{\text{meters}} = 1000 \times \frac{R + G \times 256 + B \times 256^2}{256^3 - 1}$$

4.3.5 Post-Processing

Once the video recording finishes, a post-processing step generates videos from the collected data and uses the instance segmentation maps to calculate pixel-perfect bounding-boxes, similar to the method present in Pathiraja *et al.* [2024], from pedestrians and vehicles (which are labeled according to their class such as bus, car, truck, etc). Bounding-Boxes from objects with fewer than fifty visible pixels are discarded as they may represent highly occluded or distant objects. After the post-processing is complete, the framework repeats the generation process until the number of videos generated matches the number of user-defined number of videos for the dataset. Figure 7 shows an example of data that can be generated using our framework.



(a) RGB image **(b)** Bounding-Boxes **(c)** Instance Segmentation **(d)** Depth Map

Figure 7. Example of output data provided by the generator

[10]Available at:https://carla.readthedocs.io/en/0.9.15/ref_sensors/#depth-camera. Accessed on: Mar. 17, 2025.

5 A novel dataset for smart cities applications

In this section, we present a novel synthetic dataset created to demonstrate the ENDLESS framework capability of generating large and automatically annotated datasets. We describe the dataset and provide further analysis of the data distribution and dataset parameters to provide insights about the generated data.

5.1 Dataset Description

To demonstrate the ENDLESS capability, we used the framework to create a substantial synthetic dataset, consisting of 378,751 frames distributed across 244 HD videos recorded at a frame rate of 30 FPS. Figure 8 contains image samples collected from the generated dataset. Notice the diversity of environments, viewpoints, and environmental conditions obtained by using the framework.

The final dataset was made composing six mini-datasets (hereafter referred to as "minisets") generated on different days using a consumer-grade computer equipped with an RTX 3090 GPU. Notably, the final dataset is expandable, as users can use the ENDLESS framework to generate additional minisets to extend the dataset size. Furthermore, as mentioned in Section 4.3.1, the additional user-generated minisets can be customized to allow greater control over the generated data.

To better demonstrate our framework features, all minisets were generated with all the weather, times of day, and city maps enabled. The only distinction regarding their settings is the maximum number of pedestrians and vehicles and the number of videos to be generated.

5.2 Dataset Analysis

In this section, we analyze the generated dataset to have better insights into the data distribution and verify the ENDLESS data generation. To achieve that, we analyzed the data distribution of the 6 generated minisets and also of parameters such as weather, time of day, and city maps from the final dataset.

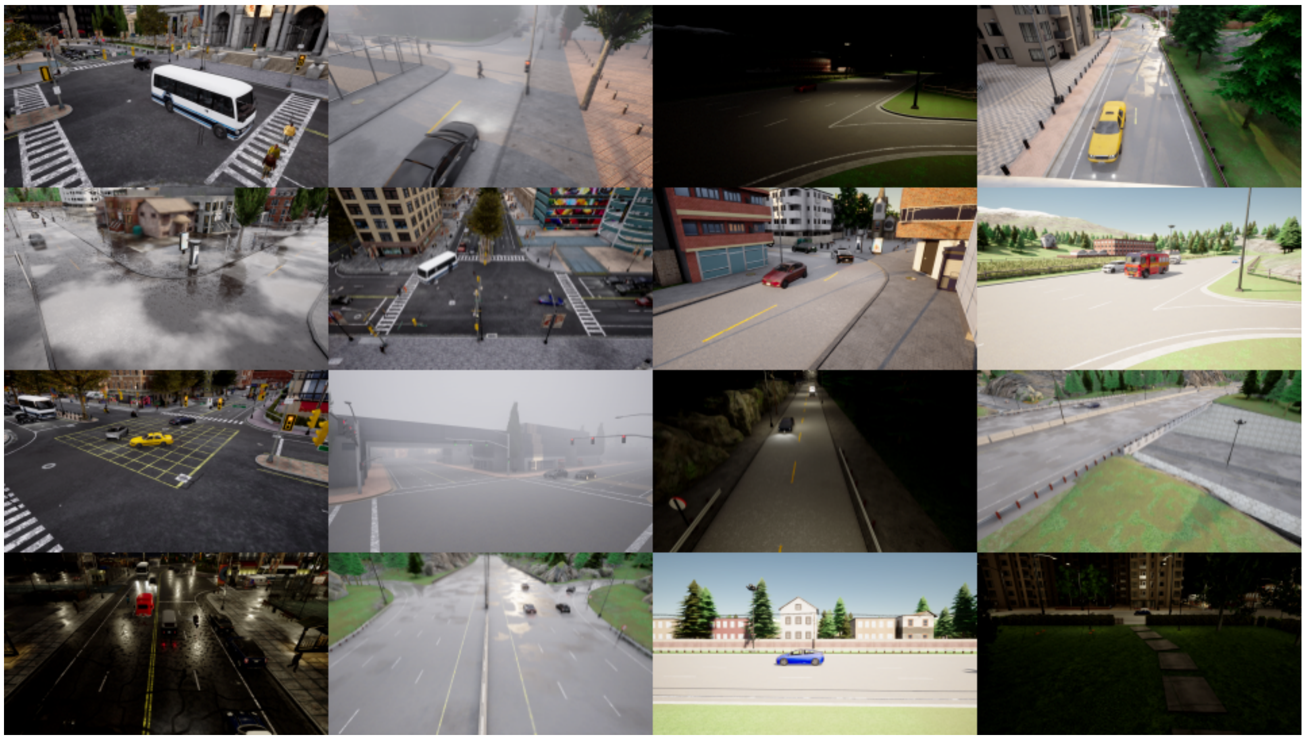


Figure 8. Samples from the dataset images

Table 2. Minisets specifications

Miniset	#Videos	#Frames	Max Vehicles	Max Pedestrians
1	18	27,632	30	20
2	104	157,269	30	20
3	30	44,957	30	20
4	40	64,050	100	60
5	30	46,840	100	60
6	22	38,003	150	100
Total	244	378,751		

In Table 2, we present the data distribution for the six minisets along with their target number of vehicles and pedestrians. The results evidence that the second miniset exceeds the number of frames and videos from the other minisets by a substantial margin. This result indicates that this miniset has the highest influence on the final dataset compared with the others.

An analysis of the dataset city map distribution is presented in Figure 9. It's visible that the cities were not used evenly across the video generation. We suppose that this is due to the random city selection and order by the city map process effectuated during the scene settings generation, as mentioned in subsubsection 4.3.2.

To better analyze the environmental conditions of the dataset, we analyzed the time of day and weather distributions both independently and in combination. This distribution is presented on Figures 10, 11, and 12. From the graphs, we note that the least represented weather condition in our dataset is "*rainy*", while the most represented is "*cloudy*". It is also observed that the nighttime period has the most data, whereas the morning has the least. Additionally, it is worth noting that the three most representative environmental conditions in the dataset, which consider both the video's weather and time of day, have different weather and times of day values, namely "*clear-sky afternoon*", "*cloudy morning*", and "*foggy night*".

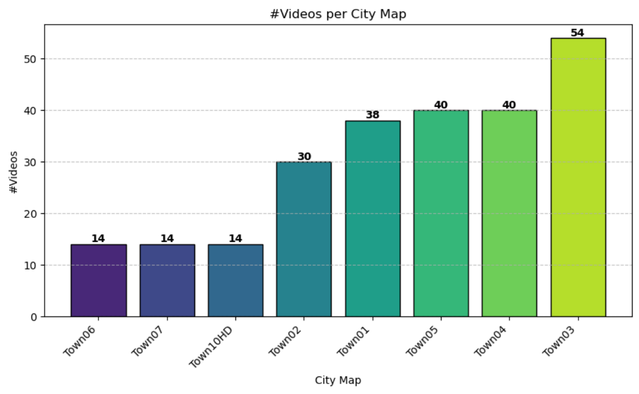


Figure 9. Dataset City Map Distribution

From the environmental graphs, we also observe that the disparity in the number of videos across classes increases as the number of classes grows. This is expected due to the random selection of weather and time of day values. This trend can be verified by comparing the time of day distribution, which has only three possible values, with the combined time of day and weather distribution, which has 15 possible values.

Algorithm 1 City Simulation

```
 1: Load city map
 2: Set weather and time of day
 3: Set TrafficManager parameters
                                                  ▷ Spawn Vehicles
 4: BatchVehiclesSpawnCommands ← []
 5: MapSpawnPoints ← Get Map Spawn Points
 6: for SpawnPoint ∈ MapSpawnPoints do
 7:     if Reached desired vehicle count then
 8:         break
 9:     end if
10:     VehicleModel ← Get Random Vehicle Model
11:     Set VehicleModel Settings
12:     Append (Spawn VehicleModel at SpawnPoint, SetAutoPilot) to BatchVehiclesSpawnCommands
13: end for
14: VehiclesList ← Execute Commands at BatchVehiclesSpawnCommands
15: Enable Vehicle Lights Control on TrafficManager
                                                  ▷ Spawn Pedestrians
16: PedestrianSpawnLocations ← []
17: for i ∈ [0..Target Number of Pedestrians] do
18:     PedestrianSpawnPoint ← Get Random Pedestrian Spawn Point
19:     Append PedestrianSpawnPoint to PedestrianSpawnLocations
20: end for
21: BatchPedestriansSpawnCommands ← []
22: PedestrianSpeeds ← []
23: for PedestrianSpawnLocation in PedestrianSpawnLocations do
24:     PedestrianModel ← Get Random Pedestrian Model
25:     Append PedestrianModel.Speed to PedestrianSpeeds
26:     Append (Spawn PedestrianModel at PedestrianSpawnLocation) to BatchPedestriansSpawnCommands
27: end for
28: PedestriansList ← Execute Commands at BatchPedestriansSpawnCommands
29: Remove speeds of unspawned from PedestrianSpeeds
                                                  ▷ Spawn Pedestrian Controllers
30: PedestrianControllerModel ← Get Pedestrian Controller Model
31: BatchPControllerSpawnCommands ← []
32: for i ∈ [0.. Number of Spawned Pedestrians] do
33:     Append (Spawn PedestrianControllerModel as child of PedestriansList[i]) to BatchPControllerSpawnCommands,
34: end for
35: PedestrianControllersList ← Execute Commands at BatchPControllerSpawnCommands
36: for PedestrianController ∈ PedestrianControllersList do
37:     Start PedestrianController
38:     Set Random Target to PedestrianController
39:     Set Pedestrian Max Speed to PedestrianController
40: end for
```

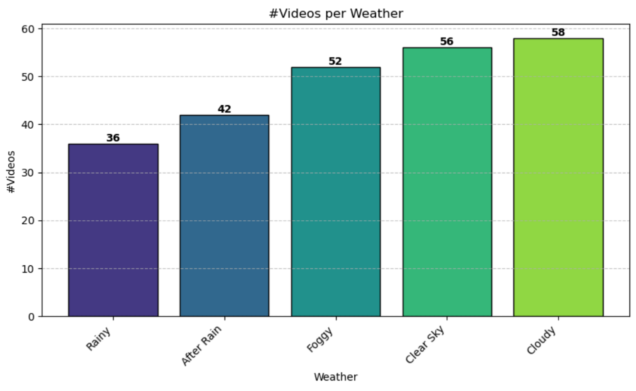


Figure 10. Dataset weather distribution

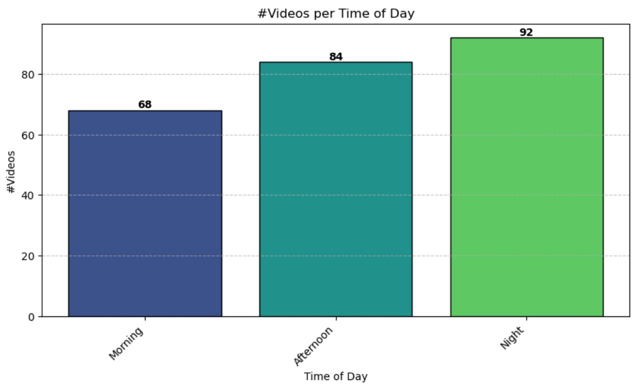


Figure 11. Dataset time of day distribution

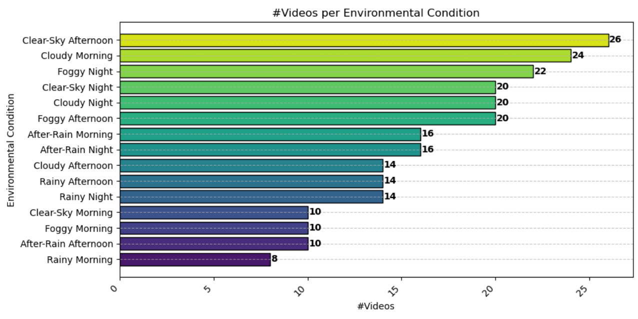


Figure 12. Dataset environmental condition distribution

6 Results and Discussion

In this section, we compare our generated dataset with state-of-the-art urban synthetic datasets. We also discuss the current limitations of our framework and present future works to improve it.

6.1 Comparison with Related Datasets

To demonstrate that our generator can create competitive synthetic datasets, we compared the generated dataset with three related synthetic datasets that are publicly available. The datasets we chose for comparison were the Synthehicle dataset [Herzog *et al.*, 2023], and the Boundless, Boundless+Digital Twin, and CARLA datasets [Turkcan *et al.*, 2024]. Further information about these datasets can be found in subsection 3.1.

We have chosen the Synthehicle dataset for comparison because its camera positioning and focus on smart city applications make it the most similar to ours. Boundless datasets

Table 3. Comparison with other datasets

Dataset	#Frames	#Views	2D Boxes	3D Boxes	Segmentation	Depth
Synthehicle	612,000	40	X	X	X	X
Boundless	8,000	1	X	X		
Boundless + Digital Twin	16,700	2	X	X		
Boundless (Carla)	22,000	1	X	X		
Ours	378,751	40	X		X	X

were selected because they were released to demonstrate the efficiency of a similar generator developed in the Unreal Engine 5[11]. We consider that comparing the resulting dataset with real-world datasets is out of the scope of this work.

The features selected to compare the datasets were: the number of frames; the number of distinct camera views; and the availability of 2D or 3D bounding-boxes, segmentation, or depth map annotations. In this work, we consider a camera view as a perspective captured from the camera based on its positioning in a given city.

The results of our comparison are presented in Table 3. Note that we consider the "Digital Twin+Boundless" dataset to have 2 distinct views as it is a merge of two datasets. Nevertheless, the camera positioning of both mimics the same real-world camera positioning. Additionally, the number of frames for the Synthehicle dataset was estimated based on the reported frame count per video and the total number of videos since the exact number was not provided in the published paper.

Our dataset contains more than seventeen more frames than the largest dataset from Boundless, made on CARLA, and more than twenty times the number of frames of the "Boundless+Digital Twin" dataset. Despite that, the Synthehicle contains approximately 61% more frames than our dataset. In terms of view count, our dataset is equivalent to Synthehicle and surpasses Boundless' by a significant margin, once they are recorded from the same viewpoint.

Regarding recorded ground truth, both Boundless and Synthehicle contain 3D bounding-box ground-truth data, which are not available on our dataset. Other than that, our dataset contains depth, instance segmentation, and 2D bounding-boxes annotations, which are also available on Synthehicle but absent on Boundless.

Regarding the features compared in Table 3, it is noticeable that Synthehicle is a larger dataset and contains a bigger range of annotations than ours. However, it is important to note that the main result of our work is the ENDLESS framework, which allows the end-to-end generation of synthetic datasets. In this case, the quantity of frames in our generated dataset should not be taken into account in order to measure quality in any way.

6.2 Limitations

The main limitation of our work is that synthetic data generated by our framework was not used yet to train and test CV models to evaluate how they can impact their metrics. However, we expect that our data can improve these models' performance as other works with urban synthetic datasets were able to do.

Another limitation is regarding the data generation. In the initial frames of the city simulation, the vehicles appear "falling from the sky" due to how the CARLA Simulator instantiates them. This unnatural event was recorded by the cameras during the construction of our dataset.

Finally, recorded videos may contain a different duration from the user-specified duration. We believe this happened due to how the CARLA simulator processes its ticks.

6.3 Future Improvements

From its current state, there are numerous potential approaches to improve our generator. Adding new annotations such as 3D bounding-boxes, Monk Skin Tones [Monk, 2019] annotations for pedestrians, and traffic-light states would increase the possible usages of the generated data.

Furthermore, adding new maps inspired by urban scenarios representative of Latin America would allow us to simulate representative data with the specificity of its cities and streets.

Finally, testing generated datasets using meaningful deep-learning models for smart city tasks would enable us to check how well these models perform on our data.

7 Conclusion

In this work, we presented an end-to-end synthetic dataset generator for smart city tasks developed using the CARLA Simulator. The generator can create automatically annotated synthetic datasets by a single script execution, mitigating the need for costly manual annotation common in real-world datasets. The generated data includes diverse scenarios with distinct camera views, weather conditions, time of day, vehicles, and pedestrians, ensuring a diverse data generation.

To demonstrate the effectiveness of our generator, we used it to generate a proof-of-concept dataset with over 300K frames and compared it with state-of-the-art related synthetic datasets. Our comparison shows that the generated dataset is paired with state-of-the-art datasets regarding frame number and number of sensors, demonstrating the generator's effectiveness.

Declarations

Authors' Contribution

A. Neto, W. Costa and V. Teichrieb contributed to the conception of this study. J. Teixeira contributed to the revision

[11]Available at:https://www.unrealengine.com/en-US/unreal-engine-5. Accessed on: Mar. 17, 2025.

of the manuscript. A. Neto and W. Costa are the main contributors and writers of this manuscript.

All authors read and approved the final version of the manuscript.

Conflicts of Interest

The authors declare that they have no competing interests.

Availability of Data and Materials

The data that support the findings of this study are available from the corresponding author upon request.

References

Adewopo, V. A., Elsayed, N., ElSayed, Z., Ozer, M., Abdelgawad, A., and Bayoumi, M. (2023). A review on action recognition for accident detection in smart city transportation systems. *Journal of Electrical Systems and Information Technology*, 10(1):57. DOI: 10.1186/s43067-023-00124-y.

Barthélemy, J., Verstaevel, N., Forehead, H., and Perez, P. (2019). Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors*, 19(9):2048. DOI: 10.3390/s19092048.

Bhat, S. F., Birkl, R., Wofk, D., Wonka, P., and Müller, M. (2023). Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*. DOI: 10.48550/arXiv.2302.12288.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., Pardo, T. A., and Scholl, H. J. (2012). Understanding smart cities: An integrative framework. In *2012 45th Hawaii International Conference on System Sciences*, pages 2289–2297. DOI: 10.1109/HICSS.2012.615.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223. DOI: 10.1109/cvpr.2016.350.

Deschaud, J.-E. (2021). Kitti-carla: a kitti-like dataset generated by carla simulator. *arXiv preprint arXiv:2109.00892*. DOI: 10.48550/arXiv.2109.00892.

Deschaud, J.-E., Duque, D., Richa, J. P., Velasco-Forero, S., Marcotegui, B., and Goulette, F. (2021). Paris-carla-3d: A real and synthetic outdoor point cloud dataset for challenging tasks in 3d mapping. *Remote Sensing*, 13(22):4713. DOI: 10.3390/rs13224713.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., *et al.* (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. DOI: 10.48550/arXiv.2010.11929.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16. DOI: 10.48550/arXiv.1711.03938.

Fabbri, M., Brasó, G., Maugeri, G., Cetintas, O., Gasparini, R., Ošep, A., Calderara, S., Leal-Taixé, L., and Cucchiara, R. (2021). Motsynth: How can synthetic data help pedestrian detection and tracking? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10849–10859. DOI: 10.1109/iccv48922.2021.01067.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349. DOI: 10.48550/arXiv.1605.06457.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE. DOI: 10.1109/cvpr.2012.6248074.

Giffinger, R., Fertner, C., Kramar, H., Kalasek, R., Pichler-Milanovic, N., and Meijers, E. J. (2007). Smart cities. ranking of european medium-sized cities. final report. DOI: 10.34726/3565.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. DOI: 10.1109/iccv.2017.322.

Herzog, F., Chen, J., Teepe, T., Gilg, J., Hörmann, S., and Rigoll, G. (2023). Synthehicle: Multi-vehicle multi-camera tracking in virtual cities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1–11. DOI: 10.1109/wacvw58289.2023.00005.

Kerim, A., Aslan, C., Celikcan, U., Erdem, E., and Erdem, A. (2021). Nova: Rendering virtual worlds with humans for computer vision tasks. In *Computer Graphics Forum*, volume 40, pages 258–272. Wiley Online Library. DOI: 10.1111/cgf.14271.

Kloukiniotis, A., Papandreou, A., Anagnostopoulos, C., Lalos, A., Kapsalas, P., Nguyen, D.-V., and Moustakas, K. (2022). Carlascenes: A synthetic dataset for odometry in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4520–4528. DOI: 10.1109/cvprw56347.2022.00498.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. DOI: 10.1109/5.726791.

Li, Y., Jiang, L., Xu, L., Xiangli, Y., Wang, Z., Lin, D., and Dai, B. (2023). Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3205–3215. DOI: 10.1109/iccv51070.2023.00297.

Monk, E. (2019). Monk skin tone scale. Available at:https://skintone.google.

Pathiraja, B., Liu, C., and Senanayake, R. (2024). Fairness in autonomous driving: Towards understanding confounding factors in object detection under challenging weather. *arXiv preprint arXiv:2406.00219*. DOI:

10.48550/arXiv.2406.00219.

Paulin, G. and Ivasic-Kos, M. (2023). Review and analysis of synthetic dataset generation methods and techniques for application in computer vision. *Artificial intelligence review*, 56(9):9221–9265. DOI: 10.1007/s10462-022-10358-3.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788. DOI: 10.1109/cvpr.2016.91.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28. DOI: 10.1109/tpami.2016.2577031.

Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016). Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 102–118. Springer. DOI: 10.1007/978-3-319-46475-6_7.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243. DOI: 10.1109/cvpr.2016.352.

Silva, B. N., Khan, M., Jung, C., Seo, J., Muhammad, D., Han, J., Yoon, Y., and Han, K. (2018). Urban planning and smart city decision management empowered by real-time data processing using big data analytics. *Sensors*, 18(9):2994. DOI: 10.3390/s18092994.

Stauner, T., Blank, F., Fürst, M., Günther, J., Hagn, K., Heidenreich, P., Huber, M., Knerr, B., Schulik, T., and Leiß, K.-F. (2022). Synpeds: A synthetic dataset for pedestrian detection in urban traffic scenes. In *Proceedings of the 6th ACM Computer Science in Cars Symposium*, pages 1–10. DOI: 10.1145/3568160.3570230.

Syahidi, A. A., Kiyokawa, K., and Okura, F. (2023). Computer vision in smart city application: A mapping review. In *2023 6th International Conference on Applied Computational Intelligence in Information Systems (ACIIS)*, pages 1–6. IEEE. DOI: 10.1109/aciis59385.2023.10367332.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE. DOI: 10.1109/iros.2017.8202133.

Turkcan, M. K., Li, Y., Zang, C., Ghaderi, J., Zussman, G., and Kostic, Z. (2024). Boundless: Generating photorealistic synthetic data for object detection in urban streetscapes. *arXiv preprint arXiv:2409.03022*. DOI: 10.48550/arXiv.2409.03022.

Yar, H., Khan, Z. A., Ullah, F. U. M., Ullah, W., and Baik, S. W. (2023). A modified yolov5 architecture for efficient fire detection in smart cities. *Expert Systems with Applications*, 231:120465. DOI: 10.1016/j.eswa.2023.120465.

Zaman, M., Puryear, N., Abdelwahed, S., and Zohrabi, N. (2024). A review of iot-based smart city development and management. *Smart Cities*, 7(3):1462–1501. DOI: 10.3390/smartcities7030061.