Delft University of Technology
Mechanical Engineering Faculty

Reducing the Sim-to-Real Gap: Lidar-Based 3D Static Environment Reconstruction

by

Joost Zaalberg

to obtain the degree of

Master of Science

at the Delft University of Technology

to be defended publicly on Friday, August 23, 2024, at 10:00.

Student Number: 4464761
Submission Date: August 17, 2024
Thesis Commitee:

Dr. Holger Caesar	TU Delft Supervisor
Dr. Son Tong	SISW, Supervisor
Dr. Michael Weinmann	TU Delft

Reducing the Sim-to-Real Gap: Lidar-Based 3D Static Environment Reconstruction

Joost Zaalberg
TU Delft

Dr. Holger Caesar
Supervisor
TU Delft

Dr. Son Tong
Company Supervisor
SISW

Abstract

This paper presents a method that shows how a lidar-based 3D static environment can be constructed from a driving scenario and is used to aid in the creation of a digital twin from that scenario. Built with limited computational resources in mind, the resulting 3D static background reconstruction method is lightweight and nonetheless up to par with similar works, proving itself to be a viable alternative to similar methods. It uses ground truth labels and open-source, out-of-the-box working building blocks to create the pipeline that filters the lidar frames, performs registration between those frames, and meshes the resulting point cloud into a 3D mesh of the static background. We performed experiments in the Siemens Prescan ADAS Simulator of a reenactment of the traffic scenario in combination with our 3D static background and measured the inference domain gap using the Bidirectional Chamfer Distance and a real-data trained object detector, comparing both the synthetic and original data. Using Prescan as the simulator, AD models van be evaluated closed-loop. This work opens up the possibilities of applying sensor domain shifts to existing data or creating reality-based traffic scenarios from existing traffic scenarios and their corresponding 3D static background of edge cases that do not, or too little, exist in real-world data.

1. Introduction

Training an Autonomous Driving (AD) model requires a lot of data, and as the amount of data increases, so does the reliability and safety of the model [15]. As the model needs to be deployed in the real world, it is commonly trained using real-world data. However, not all the required data is always available from a real-world dataset, as rare driving scenarios are often missing, such as encountering wrong-way drivers, being cut off dangerously or emergency vehicles passing through in a hurry. It would be expensive to reenact and gather this data in the real world, if not impossible. Training and validating your AD model on such data is important to ensure it knows how to act on all these long-tail driving scenarios. One way of reducing this problem is by using synthetic datasets created using a simulator. It has been shown that using synthetic datasets can improve the performance of these AD models. [11, 12]. Creating these synthetic datasets brings new challenges, including minimising the domain gap between the synthetic dataset and the real-world data, which we will address in this paper. Suppose the data can realistically be produced using a simulator. In that case, it means that more data can be produced for the training and validation of AD models, resulting in higher performance due to higher generalisation [20].

Many corner-case scenarios are underrepresented or not represented at all, which is a problem. Some corner cases can be hazardous, such as the scenarios as mentioned above. This problem can be reduced using synthetic datasets from a simulator. Using a simulator, synthetic data can safely be generated of corner cases that would otherwise be too dangerous to reenact while keeping costs down.

Another use case of simulation are sensor domain shifts. A dataset collected with one set of sensors is unsuitable for training an AD model that uses another set of sensors because it suffers a severe performance drop with such cross-domain shifts [22]. Given that vehicles and their perception equipment get more advanced, these domain shifts are inevitable, rendering older datasets collected with the previous equipment useless. By recreating the scenarios from the old dataset in simulation, the data can be recollected using the new equipment's parameters. This way, the data from the old, real-world data set is shifted into the new domain as a synthetic dataset, making it fit for training the AD models of new equipment.

The more the domain gap between simulation and real data is closed, the more realistic the simulation data becomes. When fully closed, an AD model can be trained and validated using a synthetic dataset, without compromising safety and reliability. Unfortunately, this is not yet possible,

as synthetic data still differs from real data one way or another. Currently, synthetic datasets can be used to improve an AD model trained only on real-world data [11]. This paper focuses on reconstructing the static environment from a dataset scenario using mainly lidar data. This can be used in simulation to help close the sim-to-real gap by adding a real-world-based, and therefore more realistic, 3D mesh background.

Contribution. **(1)** In this paper, we provide a method to recreate the static environment of a driving scenario. **(2)** The research shows a potentially useful method to create synthetic lidar data using a simulation of a digital twin that includes the static background and a Traffic Scenario reconstruction. **(3)** We provide a basis for developing future methods in which users can either apply sensor domain shifts to existing datasets or create novel driving scenarios without the need to create one from scratch. By modifying real-world scenarios, time and effort are reduced whilst inheriting realism from the original scene.

2. Related Work

Scenario reconstruction and simulation. Various works have been published on scenario reconstruction and simulators of these scenarios. First, the work of Ren *et al*. shows a method that only shows 3D reconstruction of a driving scenario using lidar and GPS data. It focuses on creating a robust, fusion-based and factor graph-based mapping system by combining local navigation information with GPS data [13]. Most other works have published methods that combine scenario reconstruction based on real-world data with a matching simulator to facilitate the creation of novel data by changing the sensor properties or the traffic scenarios (TS) itself. The first example of this is VISTA 2.0. VISTA represents and simulates camera images, lidar data, and event camera images from real-world datasets. It focuses mainly on creating novel views of non-altered TSs by changing the sensor's perspective, *i.e*. changing the ego vehicle pose relative to the original location or changing the location of the ego vehicle of the sensor. By utilising neural networks, the sensor's data is regenerated such that it matches the novel perspective. *e.g*., images are regenerated from a RGB-D point cloud and the lidar data is resampled such that it handles occlusions and correctly generates the typical lidar ring patterns. It allows users to test their AD models in a closed loop. Its downside is that it cannot handle scenarios that deviate too much from the original, limiting its capabilities of creating novel perspectives [1]. Second is LiDARsim, a lidar-based scenario reconstruction and simulator. It does not allow for closed-loop simulation, but simulates novel scenarios to create novel data. As input, it uses lidar point clouds with semantics. With this data, 3 libraries are created: a vehicle library, a TS library, and a corresponding 3D map library. Using the semantics, the TS is extracted and the actors are separated from the static environment. The static frames are aligned and meshed to create the 3D map to be added to the library. The points of the dynamic actors are accumulated and aligned per individual actor. After cleaning up the combined point cloud, it is meshed and added to the library. Both the 3D scenario meshes and the vehicle meshes contain the average intensity value as a mesh texture, making way for a surface material with intensity reflection properties. Using the diverse library, novel scenarios can be created by modifying existing scenarios using the resources from the library, *e.g*. by swapping out the actors of other sizes. Then, the novel scenario can be simulated to create novel lidar data, including intensities. LiDARsim does not allow for closed-loop testing of AD models [15]. Another example of a combined scenario reconstruction and simulation method is UniSim. UniSim takes recorded data from a dataset and creates a manipulable digital twin of the scenarios. It can create novel scenarios by actor modification or removal, changing sensor properties, and modification of the ego trajectory. The user can test AD models closed-loop using UniSim. It renders photo-realistic images of the novel scenes and simulates lidar sensors, using a combination of ray casting for simulating the points and a network to predict their intensities. It reports good domain gap evaluations [21]. A major downside is that it is not open-source and, therefore, unusable for the public. Contrary to the examples above, LidarDM is and leverages a diffusion model (DM) guided by the TS to create diverse, temporally consistent, and realistic lidar videos. It differentiates itself by using a DM to create novel lidar data; it does not need human assistance to create novel scenarios, reducing the need for human input. In their paper, the novel lidar data is used to increase the detection accuracy of an AD model with a generative data augmenting strategy. Its use is limited by the extensive computational resources needed to create data. [23]. Another interesting, network-based approach is EmerNeRF. Instead of using lidar data to capture the scenario, it relies solely on camera data and Neural Radiation Fields (NeRF) to capture the scene geometry, appearance, and motion. It retrieves semantics via a self-supervised bootstrapping method, which effectively separates static background and dynamic actors. Using these methods, the 3D static background and TS can be extracted from a driving scene [21]. The drawbacks of using NeRF-based scene reconstruction methods are that they do not handle occlusions in the data well and are heavily limited by the computational load required by the usage of NeRFs, as computations could take hours to days, depending on the scenarios and the resources. [10].

3. Method

Our method focuses on creating a mesh of the static environment using the point clouds of a driving scenario. We

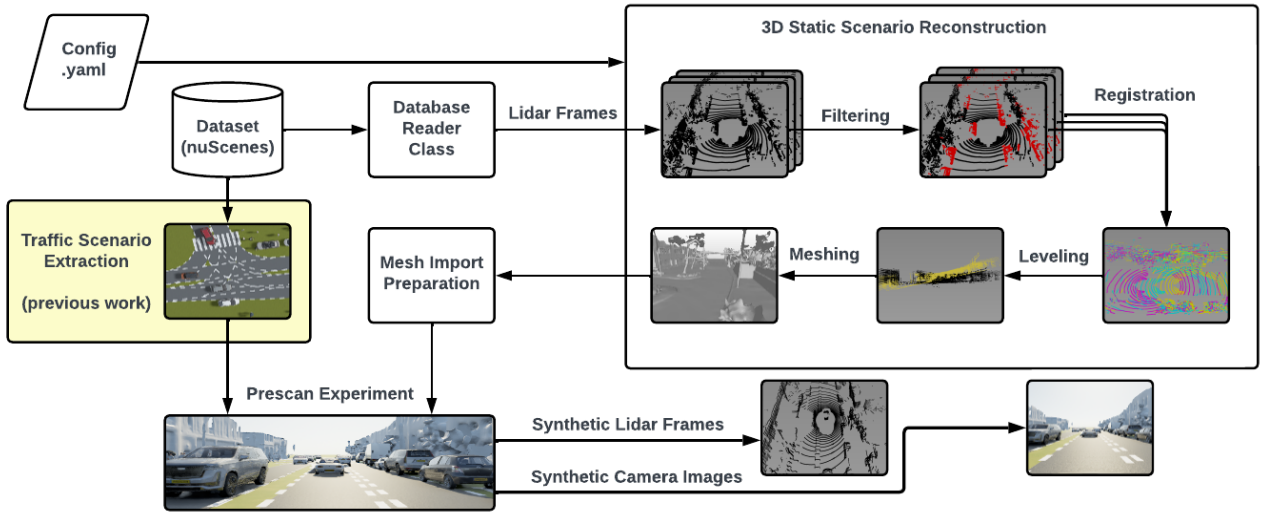


Figure 1. An Overview of the Lidar-Based 3D Static Environment Reconstruction architecture.

argue that using real data to create this static background will make the simulation more realistic. In addition, building a static background of a driving scenario (*e.g.* a city centre intersection) is labour-intensive when done manually; by automating this process, we should save time and money.

3.1. Overview of the 3D Static Scenario Reconstruction

Our method of extracting the 3D mesh from the driving scenario is written in Python [18]. An overview of our method to extract the 3D static background is shown in Figure 1. To summarise, the points of the lidar frames are filtered from noise and actors; next, the frame points are registered to each other. The whole scene is then levelled and moved to the ground plane. Finally, the 3D static scene is meshed and ready to be used as background in a simulator.

3.2. Dataset

The dataset used is nuScenes. The dataset comprises fully annotated 20s long driving scenes containing lidar and camera data, the latter has mostly been useful for visualisation purposes. The lidar nuScenes uses is the Velodyne HDL32E, collecting data with a frequency of 20Hz. [2]. The code is written such that any dataset containing the right information can be used in this pipeline: the data is read from the dataset using the separate database reader class, which feeds the information in the right format to the main pipeline. If another dataset is required, one could write its own reader class tailored to the desired dataset. For our methods to work, the reader class needs access to: i) the raw, per-frame lidar point cloud, ii) the corresponding bounding box annotations of that frame, iii) the camera images if additional visualisations are desired and iv) all the rigid transformations between the annotation frame (in nuScenes case, the world frame), the lidar frame, ego (vehicle) frame, and if needed the camera frame.

3.3. Traffic Scenario Extraction

We started with a code basis from the previous work of van Leuven *et al.*, in which they extracted the TS of the nuScenes scenarios using ScenarioNet [7]. They supplied us with the TS data extracted from each scene of the nuScenes dataset, which we used in our methods. For their research, it was sufficient to have the data on the ground plane, which unfortunately reflects in their TS dataset: it contains no z-values [17]. This had implications for our methods; the 3D reconstruction will be relocated to the ground plane. Their contribution to our work consists of data on the road network, vehicle type, and information such as size and trajectory. Their supplied code helped us import all of that into a Prescan Experiment. An in-depth description of this process will be provided in the following sections.

3.4. Filtering

First, let's look at the process of creating the 3D static mesh, summarised in the rectangle labelled as "3D Static Scenario Reconstruction" in Figure 1. The Database Reader class provides the input of this whole pipeline. The other input is the Config.yaml file, which contains configuration settings such as verbosity, in- and output paths, and filter settings. To create a static background containing as few artefacts as possible, the frame point clouds are filtered of noise and dynamic actors, shown in the Figure as red

dots. This process starts with removing the dynamic actors, which include, but are not limited to, cars, trucks, motorbikes, bicycles, and pedestrians. Note that our definition of a dynamic actor also includes parked or non-moving actors. For the removal of the dynamic actors, the Ground Truth (GT) box annotations of the dataset are used. These annotations are in the world frame, so to use this information, the points of the frame are translated, via the ego frame, into the world frame, too. Now by inflating the annotations box by a factor of 1.4, making sure to capture all points of the dynamic vehicle, these points can then be removed from the point cloud. The remaining points are then translated to the ego frame. From here, more filters are applied to reduce the noise. These are a minimum distance to the ego vehicle filter of $r > 2.7$m, a maximum distance filter of $r < 33$m, and a below-the-ground-plane filter of $z < -0.5$. The minimum distance filter's function is to remove points that reflect from the ego vehicle, the maximum distance filter removes outliers and points that have decreased accuracy, and the below-the-ground-plane filter removes points that appear to be under the ground plane due to errors or reflections from the road surface. All the filtered frames are saved for registration.

3.5. Registration

The next step is to register all the filtered scenario frames to each other. Note that our frames are currently in the ego vehicle frame. Registration is the process of aligning two or more point clouds into a common coordinate system. For this, KissICP [19] is used. As stated in their paper, their algorithm does not need any parameter tuning to function on specific lidar data, and thus, it is used as-is. Using the frames in the ego frame, the KissICP algorithm outputs the homogeneous transformations that align all the frames relative to the first frame. This returns a point cloud of the whole scene, with dynamic actors removed and its origin at the first frame's origin, which is in the ego frame. This conveniently places the origin at the same location as the origin of the TS data. However, the axes are not aligned yet. This misalignment will be addressed during the setup of the Prescan experiments.

3.6. Levelling

Because the TS data is flattened to the x-y plane by the lack of z-values, the drivable road surface of the scenario needs to be as close to the ground plane as possible, or just shy below it. A simple method was developed to rigidly transform the points to the desired pose.

Our method uses lidar segmentation labels of the lidar points to extract all the points labelled as the drivable road surface. These points are used to fit a plane using formula 1:

$$ax + by + cz + d = 0 \tag{1}$$

in which the x, y, and z are the cartesian coordinates and a, b, c and d are the parameters found from fitting a plane to the drivable road surface points. The a, b and c parameters are used in formulas 2 to rotate all scene points around the origin to make the road surface parallel to the ground plane:

$$\begin{aligned} \vec{n}_{target} &= [0, 0, 1]^T \\ \vec{n}_{plane} &= [a, b, c]^T \\ \vec{v}_{rot} &= \vec{n}_{target} \times \vec{n}_{plane} \\ \theta &= \arccos(\vec{n}_{target} \cdot \vec{n}_{plane}) \end{aligned} \tag{2}$$

In which the normal vector $\vec{n}_{target}$ is that of the target plane, which is the ground plane, and the normal vector $\vec{n}_{plane}$ is that of the drivable road plane. $\vec{v}_{rot}$ and θ are the rotation vector and angle that align the plane to the ground surface. After rotation, the scene point cloud is translated along the z-direction by $-d$ to bring the levelled surface closer to the ground plane. Note that the above plane fitting method heavily relies on the assumption that the drivable road surface is reasonably flat. In some cases, it turned out that the scene was not flat enough, rendering the scenario unusable for our method.

3.7. Meshing

The final step before exporting the mesh to a simulator is meshing the point cloud. This is done using the Neural Kernel Surface Reconstruction (NKSR) method developed by NVIDIA. The open-source surface reconstruction and meshing algorithm is highly generalizable and requires a GPU, resulting in a fast and out-of-the-box solution, as it did not need any parameter tuning. Some examples of its performance on selected regions of the dataset can be seen in Figure 2.

3.8. Mesh Import Preparation

A manual, intermediate step is required to import the created mesh into Prescan. A deep dive into these necessary steps to convert the mesh file into a file structure usable by the Prescan simulator can be found in the Appendix, section A. This information was not readily available and might be of interest to those who want to reproduce this step themselves. These steps are necessary for Prescan to be able to find the created 3D mesh and to ensure the synthetic lidar sensor works properly.

4. Experiments

This section evaluates the performance of our proposed method using experiments executed in the Siemens Prescan [14] simulator. Prescan is a software suite that enables

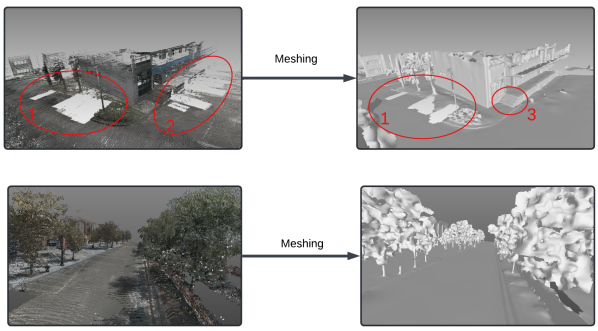


Figure 2. Two close-ups of the point-to-mesh process. On the left are two points clouds, on the right have they been meshed. In the upper example, a street corner with a building is seen. Note that in the point cloud, there are holes at ovals 1 and 2. The rectangular holes in the point cloud result from removing parked cars, as removing the points within the slightly inflated bounding also removes the ground points. The meshing algorithm is able to fix the holes at number 2 but not at number 1. Number 3 is a good example of how the meshing algorithm is able to produce great detail: all the individual steps of the stairs are there. Below is an example of how the algorithm handles vegetation. Though the meshing algorithm does not produce great details in the leaves of the trees, it does create some irregular tree-like structures.

the development and validation of ADAS systems [16]. Its prime interface is Matlab [5] Simulink-based, which allows for the transfer of data extracted from the various scenes, such as the TS, the ego vehicle's trajectory and the 3D mesh created in the previous section. Prescan features various physic-based sensors, such as lidar, radar and cameras. Almost any parameters of these sensors can be set to values that match the dataset vehicle car. Not only is this key for applying future sensor domain shifts, but it is also key for validating our methods because we need to set it to the exact parameters of the original dataset. To ensure that the sensor domain gap is as small as possible, the lidar sensor's parameters are set to the same value as the nuScenes ego vehicle's. By combining the TS data (that includes the ego vehicle) with the previously created 3D static scene, we have effectively created a digital twin of the scenario. By keeping the TS the same, we can have a frame-to-frame comparison between the original nuScenes lidar data and the synthetic lidar data from the simulator. We will now dive into some details on the simulation setup and how the complete, *i.e.*, including fore- and background, simulations are created.

4.1. Simulation setup

Software. We conducted our experiments using Siemens Prescan, version 2403 [14]. Using the Matlab Simulink version 2021b [5] enabled the process of loading all the data described in the previous section.

The Experiments. An experiment consists of a digital twin of a single driving scene from the nuScenes dataset, reenacting it from the beginning to the end. In total there have been 11 experiments performed, all of which contain about 400 lidar frames. The inputs of the experiment are two-fold: the first input is the TS, and the second input is the 3D static environment mesh. Let's dive a little deeper into what information is extracted from the TS data.

Traffic Scenario. Using Leuven's methods [17] and their code as a basis to import the TS, the first piece of data extracted from the TS is the road network. The available road network in the TS data covers an area proportionate to that of a whole neighbourhood, so we have limited the road import to just those roads that reach within the cartesian limits of the 3D mesh, which most of the time contain just an intersection or two. Secondly, the road network is enhanced with features like bike lanes, crosswalks, stop signs, and speed bumps, among others. Up until now, the imports are visualised by the simulator instead of using the information from the 3D mesh. The third step is importing the vehicles and pedestrians present in the experiment's TS. Our choice of picking a vehicle differs from the original method of van Leuven's work, as their vehicles were picked from the library randomly. In our method, for every vehicle (such as a bus, truck, car, etc.) annotated in the dataset, a model of the same type of vehicle is picked from the Prescan vehicle library by similarity with the annotation's dimensions. The metric used for measuring the similarity in size is by comparing the $l + h$ of the annotation's bounding box (BB) and matching it to the closest vehicle's $l + h$ in the Prescan library. The L1 norm has been picked over the L2 norm because it is less sensitive to the fact that vehicles are always longer than they are high, giving equal importance to the difference in height of the original and simulated vehicle and their difference in length. Some BBs from the TS are likely wrong, e.g. a lot of lengths are below a meter. For all those vehicles smaller than the smallest car in the library, another random vehicle has been assigned instead. Using the nuScenes camera footage, it was validated that traffic cones, bikes or bicycles have never been mistaken for cars. For pedestrians, motorbikes, and bicycles, a same-type model from the Prescan library is picked randomly because their dimensions are so similar. After the actor's models have been picked, the corresponding trajectories of the imported objects, including the ego vehicle, are assigned. The objects's trajectories exist in the TS dataset out of discrete points; a smooth path along these points is calculated, and this path is assigned to the object's trajectory. This changes the static scene into a dynamic scene where all the actors move smoothly and behave like in the nuScenes dataset.

3D Mesh. Next, the 3D static background mesh is imported. The 3D mesh is plain white after the mesh import preparation, due to technicalities in the software described

in the Appendix, section A. The alignment problem mentioned in 3.5 is addressed by rotating the mesh around the z-axis by a yaw value found in the dataset *i.e.* the yaw value of the 1st lidar from to the world frame. This aligns the background mesh perfectly with the TS coordinate frame.

Sensors. The sixth and last step of setting up our experiment is configuring the sensors, starting with the front camera. Its pose relative to the car is saved per scenario in the nuScenes dataset, as it can slightly differ per used car, *i.e.* the data is collected in both Boston and Singapore. The sensor poses on the different cars are close, but slightly different. Furthermore, the following camera parameters are found on the nuScenes website [8] and configured in the Prescan: Focal length, resolution, image sensor dimensions, and its update frequency of 12Hz. Next is the Lidar sensor. The type of simulated lidar sensor is a non-physic-based lidar sensor, which implies, in our case, that it does not return intensity values, only the x, y, and z coordinates of the points. The lidar's pose is also set correctly according to the nuScenes data. Then, the lidar's properties are set to match the Velodyne HDL32E's properties, that is conveniently preconfigured available within Prescan. Its frequency is set to 20Hz. Now, the experiment is ready to run.

4.2. Evaluation Metrics

The performance was measured by comparing the same lidar point cloud frames of the original nuScenes data and the synthetic data created by the simulation. The main metric used for this is the mean bidirectional chamfer distance (CD), shown below in equation 3 [22]:

$$\begin{aligned} CD(\hat{G}, G) \ &= \frac{1}{|\hat{G}|} \sum_{\boldsymbol{x} \in \hat{G}} \min_{\boldsymbol{y} \in G} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ &+ \frac{1}{|G|} \sum_{\boldsymbol{y} \in G} \min_{\boldsymbol{x} \in \hat{G}} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 \end{aligned} \tag{3}$$

in which $\hat{G}$ and G are the two point clouds that are compared and $\boldsymbol{x}$ and $\boldsymbol{y}$ are the respective points in these point clouds. The $\min_{\boldsymbol{y} \in G} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ and $\min_{\boldsymbol{x} \in \hat{G}} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2$ parts are the individual CD for a point to the closest point in the other point cloud. For calculating the CD's, we have used the Python module `chamfer_distance` [9]. It not only provides the average bidirectional chamfer distance (useful for calculating a scene-average value) but also returns the per-point CD, which is very useful for visualising the results. This helps to evaluate the fore- and background points separately, and is necessary to create visualisations that point out which areas of the simulation perform well and which do not. Only the points in the range $2.7 < r < 10$ m have been evaluated so that the results can be compared to those in the work of Rajendran *et al.* [12]. A value of 2.7 was picked, so the points hitting the ego car were not included in the evaluation. The second method that is used to quantify the quality of the simulation is a comparison between the detections of a Point Pillars [6] object detector, implemented using the python library MMDetection3D [4], on the real and synthesised lidar data. By comparing the detections of the real and synthetic data point cloud, and treating the detections on the real dataset as the ground truth, the similarity between the detections of both clouds can be analysed. For this, we must first determine which BB of the synthetic data correspond to which BB of the real data. This is done using a greedy matching algorithm that matches the two BBs that have combined the highest Intersection over Union (IoU) value, if greater than the IoU threshold. Using the statistics of the matched and unmatched BB, a confusion matrix can be constructed and used to calculate the mean recall and precision scores over all frames. Because a translation error in the data was suspected, we also present the Average Translation Error (ATE) of the matches. Note that because the simulation is flattened to the ground plane, the z-values of the BB are omitted too. All the IoU and ATE calculations are done with the 2d BBs on the ground plane. Also note that the goal is not to measure the performance of the object detector, but to get an indication of how similar it performs on the real and synthetic data. If the detector creates similar results, one cloud can be used as a proxy for the other, meaning that the synthetic dataset can be used to test our perception models on long-tail corner cases. Unfortunately, it does not mean it can be used for training, as this does not necessarily mean it also has a small training domain gap. By measuring the scene's bidirectional CD value and evaluating the performance of an object detector on both the real and synthetic datasets, the inference domain gap is measured [12].

To better understand how the static background performs independently, evaluating it separately from the foreground is crucial. This is done in a slightly new experiment setup. Lidar point clouds are collected again in the same experiment, but this time without any objects but the ego vehicle. This creates a lidar point cloud of just the static scene. The challenge is to compare this point cloud to the real nuScenes data, but without the points caused by objects. The second challenge is that objects in the original point cloud data cause a lidar shadow, in which no points exist. These two challenges are solved by removing all the points within the GT BB and all points in the lidar shadow behind those GT BB in both the real and synthetic data. This creates two, partly incomplete point clouds that contain only background points, over which the average BiCD is calculated. An example of this can be found in the Appendix, in Figure 9. The number of points in the complete simulation minus the number of points in the background-only simulation yields roughly the number of points reflected by objects in the complete simulation, called the foreground.

Scene	Mean CD Real-to-Sim [m]	Mean CD Sim-to-Real [m]	Median BiCD [m]	Mean BiCD [m]	Mean BiCD Background [m]	Mean BiCD Foreground [m]	Foreground points [%]
0131	0.12	0.12	0.24	0.24	0.22	0.73	7
0132	0.09	0.09	0.17	0.19	0.16	0.62	17
0164	0.14	0.13	0.29	0.28	0.21	1.1	25
0251	0.15	0.18	0.30	0.34	0.21	0.51	37
0398	0.30	0.33	0.59	0.64	0.36	0.87	48
0562	0.18	0.23	0.36	0.40	0.33	0.63	20
0663	0.19	0.21	0.28	0.35	0.29	0.74	18
0769	0.16	0.16	0.28	0.33	0.25	0.75	26
0778	0.27	0.10	0.23	0.36	0.32	0.80	13
0952	0.09	0.12	0.20	0.21	0.20	0.59	8
0955	0.14	0.18	0.29	0.32	0.32	0.47	2
overall	**0.17**	**0.17**	**0.29**	**0.34**	**0.26**	**0.69**	**18**

Table 1. The Chamfer Distance (CD) and Bidirectional CD (BiCD) values for all the evaluated scenes. Note how the median value is lower than the mean value, showing that outliers in the CD values cause a higher mean value. In the most right three columns, the mean CD values are separated into those from the background (*i.e.*, the static scene) and those from foreground points (*i.e.*, (dynamic) objects). The percentage foreground points are shown in the far-right column, being a percentage of the total amount of points in the complete simulation.

Based on these numbers, the BiCD of the foreground can be calculated using formula 4, which is derived from the fact that the weighted average of the BiCD of both the back- and foreground points add up to the average BiCD:

$$\mathrm{BiCD_{fg}} = \frac{\mathrm{BiCD} - \mathrm{BiCD_{bg}}(1 - f_{fg})}{f_{fg}} \tag{4}$$

in which the subscripts bg and fg stand for back- and foreground. The BiCD without subscript stands for the mean BiCD of all points in the complete simulations with objects compared to the original nuScenes data. f_{fg} is the fraction of foreground points in the complete simulations.

4.3. Results

Running the Experiments. The algorithm creating the static mesh was run on a laptop with 64GB RAM and a laptop version NVIDIA RTX A4000 GPU, which is inferior to the desktop version. Creating the meshes using our pipeline took an average of 5 minutes per scene, each using around 8 million points. The Prescan experiments were run using the same laptop, having an average runtime of 4 minutes per scene.

Chamfer Distance. The results of the CD calculations on the experiments' point clouds are summarised in Table 1, in which for every experiment, the mean CD of the real nuScenes data to the synthetic data and the mean CD of the synthetic data to the real nuScenes data is given. The next two columns to the right are the corresponding median and mean bidirectional Chamfer Distance (BiCD). The mean BiCD values are separated into the respective fore- and background values in the three most right columns. Appendix B shows a more elaborate overview of the CD results on a per-scene per-frame basis. In Figure reffig:hist and Table 2, more statistics are shown on the distribution of the CD values of the individual points.

Object detector. The results of comparing the detections of the real and synthetic data are shown in Table 3, and an illustrative example of this is shown in Figure reffig:BEVdets.

5. Discussion

In this section, an analysis will be performed on the results shown in the previous section. The goal here is not to present an optimised solution but rather to identify and show where the digital twin underperforms and how it could be improved.

5.1. Interpretation of Results

Evaluating the CD values presented in Table 1, the digital twin seems to perform on par with the related work of Rajendran *et al*. [12]. However, by closer inspection of the per-frame average CD graphs in Appendix B Figure 7, it is clear that some features of the digital twin are underperforming. These graphs give a good insight into when the digital twin is performing well and when it is not. The peaks in the per-frame-average CD scores mark these moments of failure. Once identified, these frames are closely examined to determine which areas are underperforming compared to the other, using a visualisation in which points are coloured by their respective CD value, as seen in Figure 5.

There are several improvements to make in our method.

CD [m]	<0.05	<0.1	<0.2	<0.3	<0.4	<0.5	<0.6	<0.7	<0.8	<0.9	<1.0
Real-to-Sim Fore- and Background [%]	47.4	65.5	81.8	88.8	92.4	94.3	95.4	96.1	96.7	97.1	97.5
Sim-to-Real Fore- and Background [%]	44.2	62.4	79.4	87.2	91.4	93.8	95.2	96.1	96.8	97.3	97.7
Real-to-Sim Background [%]	49.2	68.8	85.7	92.6	95.8	97.4	98.1	98.6	98.9	99.1	99.2
Sim-to-Real Background [%]	44.6	63.9	81.6	89.6	93.7	95.9	97.1	97.8	98.3	98.7	99.0

Table 2. Percentages of the total amount of Real and Simulated lidar points below a certain Chamfer Distance (CD) value. These statistics show how the majority of points lie closely to their corresponding points cloud. Again, the background performs better than the simulation overall.

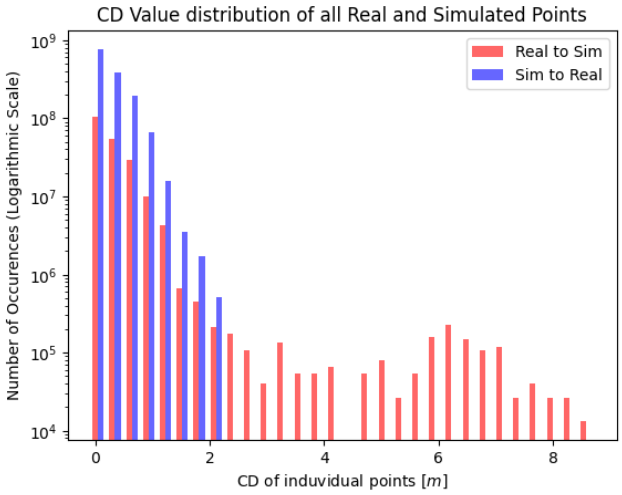


Figure 3. A histogram of the Chamfer Distance (CD) values of all evaluated points with a radius of 10 meters around the lidar sensor for the real and synthetic data. Note the logarithmic scale on the y-axis and how there are little to no points of synthetic lidar points with CD $> 2.5\ m$. The real data points contain CD values between 0 and 9 meters. The simulation's points behaviour can be explained by how mesh surfaces are created. A mesh is created where there have been a lot of points originally, creating the surface the simulation lidar can reflect on. In this Figure, scene-0164 has been excluded. The 'spoof', as described in Appendix B Figure 7c, causes individual CD values to rise up to 35 meters, which should not be possible, and hence it has been excluded for this figure. The spoof does not affect the results shown in Table 1.

IoU Treshold	Precision	Recall	ATE [m]
0.5	0.45	0.40	1.9
0.3	0.59	0.51	2.6
0.1	0.72	0.61	4.9

Table 3. A comparison of the detections using the real and the synthetic data. The values shown are the mean over all frames in the dataset. Note that the goal is not to compare the performance of the object detector, but to compare the similarity of their outputs. The IoU threshold is the threshold at which two BB from real and synthetic data are considered a match. The used IoU thresholds are quite low (usually a IoU value of 0.5 is considered a match) because a translation error in the TS data is suspected, hence why the Average Translation Error (ATE) column is added to provide additional data to underpin this hypothesis.

First, there is the replacement of vehicles that do not seem optimal; see Figure 4 and 6. The problem seems to be twofold: First, there seems to be an error in the algorithm placing the vehicles. A closer inspection of the vehicle placement algorithm revealed that the vehicle's origin in Prescan is in the middle of the rear axle on the ground plane. The algorithm places these origins in the middle of the GT BB, effectively positioning the cars with an offset towards the front of the car. This results in the rise of the ATE and causes a lower IoU between vehicle BB that should've been a match. Secondly, As mentioned in Figure 6's caption, the 'simulation vehicle gap' does not provide enough cars in the $l + h$ range between 8 and 13 meters. Both vehicle errors can cause a seriously large shadow miscast, resulting in the rise of the CD value. Also, the lower-left cluster of points shows the many incorrect vehicle annotations in the TS dataset. Referring to Figure 9 in the nuScenes' publication [2], it can be seen that their annotations do not contain these strange values. In Figure 4 and Table 3, it is clearly shown that there exists a translational error in the vehicle's position, too.

Another contribution, but a lot smaller, are artefacts in the 3D mesh. Though sometimes creating inaccuracies, these artefacts do not significantly increase the average CD value. What is more worrying is where these occur; the CD metric does not capture this. For example, the artefact in scene-0251, as seen in Figure 5b and in Appendix B, Figure 8b,

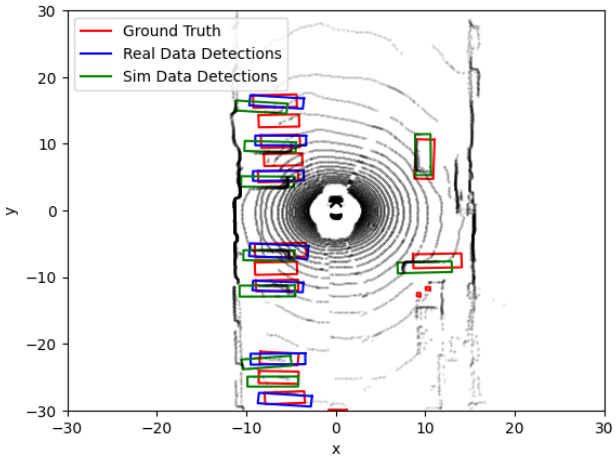


Figure 4. A plot in BEV perspective showing the synthetic point cloud of scene-0164 frame 380. The plot is overlaid with the BB of the ground truth and the BB of the detections on the real and simulation data. This frame has been picked because it clearly shows the translation error. In the scene, the ego car drives through a street with parked trucks. In this frame, the precision is 0.50, the recall is 0.71, and the ATE is 5.1m.

is in the middle of the road and on the ego vehicle's trajectory. If this data were to be used in training an AD model, it would be taught to continue to drive into an obstacle. This is, of course, unacceptable, and these kinds of artefacts should be addressed. This artefact has likely been created due to unremoved vehicle object points, emphasising the importance of inflating the GT bounding box during filtering large enough so that this cannot occur.

5.2. Comparison with Related Work

Our method achieves comparable mean BiCD scores to existing methods in the literature, for example, the work of Rajendren *et al*. [12], who has reported a scene average biCD of $0.3m$. Interestingly, their foreground scores are lower than their average CD, whereas, in our method, the foreground often causes a rise in CD scores. Our background seems to outperform their work; however, a direct comparison is not feasible due to differences in the reporting method. While our simulation does not reach state-of-the-art performance, it does provide an alternative, simplistic way of achieving similar results using limited computational resources [21, 23], especially if the problems related to the foreground are resolved.

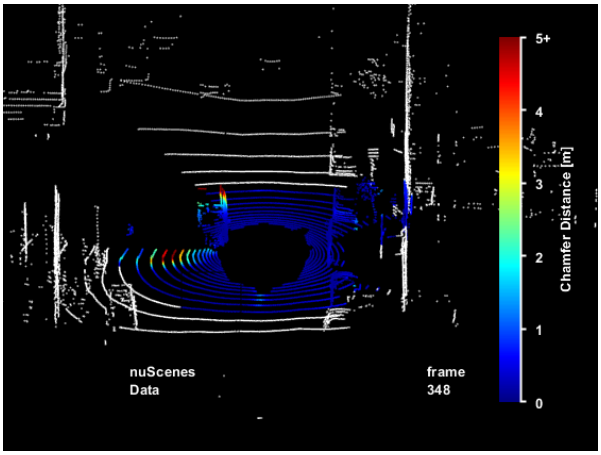
6. Conclusion

This paper shows a computationally lightweight method of creating a 3D static background mesh from driving scenes, which seems to be up to par with comparable works. We also show how we made a complete digital twin of the corresponding driving scenario in Prescan and used it to create lidar data. After evaluation, we must conclude that the complete simulation underperforms, mainly because the vehicles are not replaced accurately enough. Nevertheless, it shows great potential when its problems are addressed and should serve well as a basis for future research. If only the 3D background mesh were required, this part could easily be used independently for further research.

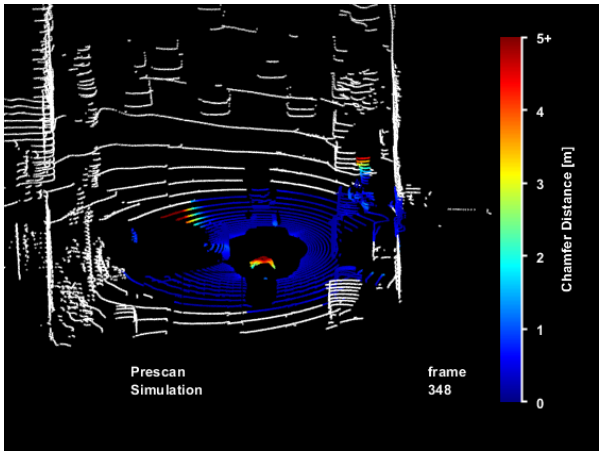
Limitations. A limitation of our method is the reliance on GT labels, which may not easily be generalised to other, more generic or in-house collected data. Although using GT labels has been great for focusing on creating the 3D mesh, it should not be a dependency for a final product.

Future Work. To improve the static background reconstruction, future research should include methods to ensure no dangerous obstacles on the ego car's trajectory, as is seen in Figure 8b. Also, a very informative figure that could be a good performance indicator on the various parts of the point cloud, is a figure that shows the average CD value per lidar segmentation label category, such as vehicle, tree, drivable road, etc. The nuScenes data includes lidar segmentation labels, which could be used to calculate the real-to-sim CD of the categories. Another future improvement could be to include a colour texture to the mesh. This way, the use case of the 3D mesh can be expanded to include photorealistic camera images. To improve the quality of the lidar data, a mesh texture of the average lidar point intensity could also be created. This could aid in making a surface material with intensity properties that could be used by Prescan's intensity-returning, physics-based lidar sensor. To improve the scenario's simulation as a whole, future research should focus on improving the methods of capturing and replacing the TS. This should include verification between the real data's GT bounding boxes and the GT bounding boxes extracted from the simulator to ensure the object's positions and dimensions are correct. Another improvement could be made by expanding the TS and the 3D mesh to include the 3rd dimension. By including the z values, a more accurate resemblance of the 3D mesh can be created in which the vehicles drive over the 3d mesh instead of a simulated road. This expands the set of usable scenes by those scenes that are not flat, removing the need to level the scene to the ground plane. Another improvement in the TS can be made by creating a complete vehicle library that covers the Prescan vehicle library gap and has more diversely dimensioned vehicles.

Acknowledgements. Special thanks to the work of Leuven *et al*. for providing us with the Prescan code base and the TS extraction. Thanks to Jasper van Leuven personally, he was always available to debug, brainstorm, and bring methods to the next level. Hamid Abdolhay was extremely helpful in solving all the problems regarding the Prescan software. As a Siemens employee working in the Prescan maintenance



(a) The original nuScenes data. The inaccurate replacement of the left truck vehicle, whether it is the vehicle size or location, causes a miscast shadow.



(b) The Prescan synthetic data. Note how the location and size of the left truck vehicle are off, also the vehicle seems to be too small. An artefact in the middle of the road causes the high CD value points near the ego vehicle.

Figure 5. Frame 248 of scene-0251, two point cloud visualisations of the real and simulation data of which evaluated points are coloured by their corresponding CD value. This is one of the many examples of how an incorrectly replaced vehicle causes a lidar shadow with no neighbouring points. In Appendix B Figure 8, the corresponding camera images are shown for comparison's sake, as well as the background-only comparison in Figure 9

and development team from the same office, his quick responses and solutions never disappointed. Besides the technical support, there has always been unconditional loving support from my partner Anouk, giving me all the much-needed space. Special thanks to Bumi, who was always happy to see me and eager to take me for a walk when I needed it most. A final thanks to all those who supported me, including my family, friends, supervisors, and doctors, every time I got delayed by another medical situation.

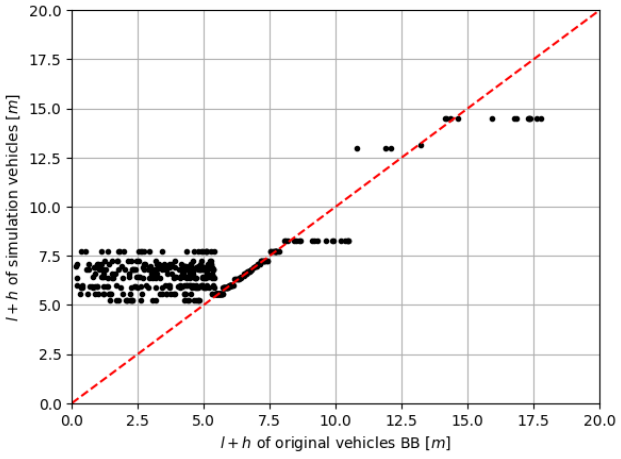


Figure 6. A graph of the $l + h$ of the original BB vs the $l + h$ of the imported vehicles. The points in the left cluster are all the vehicles with presumed wrong BB dimensions, smaller than the smallest car in the library, which has been discussed in section 4.1. These have been assigned a random car vehicle instead. Note the 'Prescan vehicle library gap' that runs roughly between the simulation vehicle's $l + h$ of 8 and 13 meters.

References

[1] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles, 2021. 2

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020. 3, 8

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 12

[4] MMDetection3D Contributors. Mmdetection3d: Openmmlab next-generation platform for general 3d object detection, 2020. 6

[5] The MathWorks Inc. Matlab, 2021. 5

[6] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds, 2019. 6

[7] Quanyi Li, Zhenghao (Mark) Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Advances in Neural Information Processing Systems*, 36:3894–3920, 12 2023. 3

[8] Motional. Nuscenes - data collection. Accesed on 01-08-2024. 6

[9] Otaheri. Chamfer distance for pytorch, 2022. GitHub Repository. 6

[10] AKM Shahariar Azad Rabby and Chengcui Zhang. Beyondpixels: A comprehensive review of the evolution of neural radiance fields, 6 2023. 2

[11] Vickram Rajendran, Chuck Tang, and Frits Van Paasschen. Improving rare classes on nuscenes lidar segmentation through targeted domain adaptation, 2023. 1, 2

[12] Vickram Rajendran, Chuck Tang, and Frits van Paasschen. Analyzing synthetic datasets through the training and inference domain gap, 6 2023. 1, 6, 7, 9

[13] Ruike Ren, Hao Fu, Hanzhang Xue, Zhenping Sun, Kai Ding, and Pengji Wang. Towards a fully automated 3d reconstruction system based on lidar and gnss in challenging scenarios, 2021. 2

[14] Siemens Digital Industries Software. Prescan, 2024. 4, 5

[15] Manivasagam Sivabalan, Wang Shenlong, Wong Kelvin, Zeng Wenyuan, Sazanovich Mikita, Tan Shuhan, Yang Bin, Ma Wei-Chiu, and Urtasun Raquel. Lidarsim realistic lidar simulation by leveraging the real world - 2020, 2020. 1, 2

[16] Siemens Digital Industries Software. *Simcenter Prescan Manual, version 2403*, 2024. 5, 12

[17] Jasper van Leuven. A data augmentation pipeline: Leveraging controllable diffusion models and automotive simulation software, 2024. Master Thesis, in collaboration with Siemens. 3, 5

[18] Guido Van Rossum and Fred L. Drake. Python 3 reference manual, 2009. 3

[19] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way, 9 2022. 4

[20] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 626–643, 2018. 1

[21] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision, 11 2023. 2, 9

[22] Bo Zhang, Xinyu Cai, Jiakang Yuan, Donglin Yang, Jianfei Guo, Xiangchao Yan, Renqiu Xia, Botian Shi, Min Dou, Tao Chen, Si Liu, Junchi Yan, and Yu Qiao. Resimad: Zero-shot 3d domain transfer for autonomous driving with source reconstruction and target simulation, 9 2023. 1, 6

[23] Vlas Zyrianov, Henry Che, Zhijian Liu, and Shenlong Wang. Lidardm: Generative lidar simulation in a generated world, 2024. 2, 9

A. Mesh Import Preparation

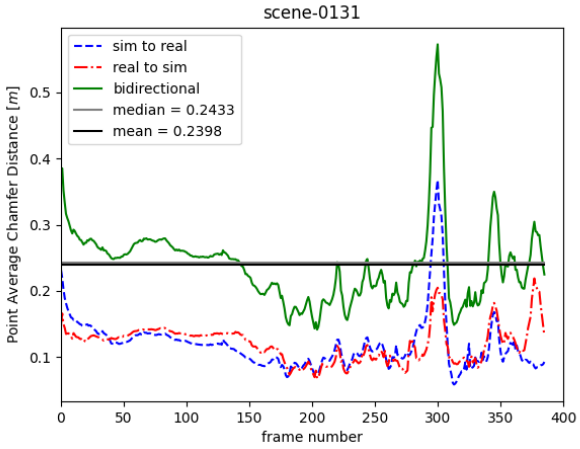
This section goes into some technicalities of creating an importable and functional mesh for the Prescan simulator that is only of interest to those familiar with the Prescan software, who are interested in reproducing this step.
After saving the mesh at the end of the pipeline as a .ply file of vertices and faces with normals, it is imported into Blender (version 3.1) [3], which must be launched from the Prescan Terminal. A material must be assigned to this mesh to enable the Prescan to use the mesh as a reflective surface. Any material will do, so just a material with a regular base colour is used for simplicity's sake. If a coloured mesh were to be desired, then UV maps and textures would be assigned. Unfortunately, the complex shapes of our static scene meshes prevented us from finding a method that enabled us to create and assign these. The final step in Blender is to export it as a .psscene file using the Prescan Blender plugin. The next step is to import the .psscene file into the Prescan Model Preparation Tool (MPT). This tool imports models of scenery, vehicles, and other objects, and assigns specific properties to them *e.g.*, reflectance, dynamics, and much more. These files are then saved in the "GenericModels" folder, as stated in the manual [16], and also need to be saved in the "UDLibElements" folder and the experiment file's folder, which is not stated in the manual. Performing these steps ensures that the scene model can be found in the Prescan model library and that the synthetic lidar sensor works properly. When using the MPT tool, the mesh can not keep its vertice colours, so it reverts to a white appearance. There exists a workaround for simpler meshes where UV maps and textures define the appearance of the mesh, but we have not found a method that works for such complex shapes as 3D static background meshes. This is not a problem when synthesising a lidar point cloud containing only x, y, and z values. It is, however, visually less pleasing.

B. Supportive Visualisations

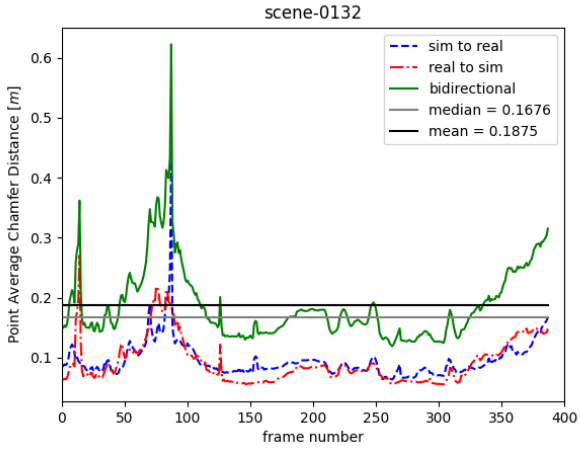
In the following pages, some figures are presented which contain supportive visualisations of the experiments's results. An elaborate overview of the results is given in Figure 7, where the graphs show for every evaluated scene the frame average CD values.

In Figure 8, a pair of camera images are shown in addition to the information given in Figure 5

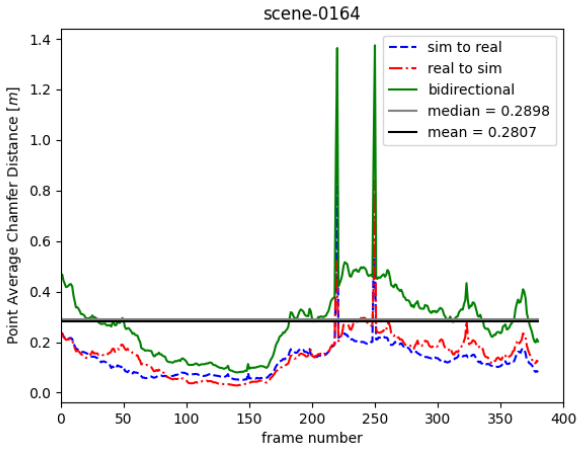
Figure 9 shows how the CD values of only the background points are calculated.



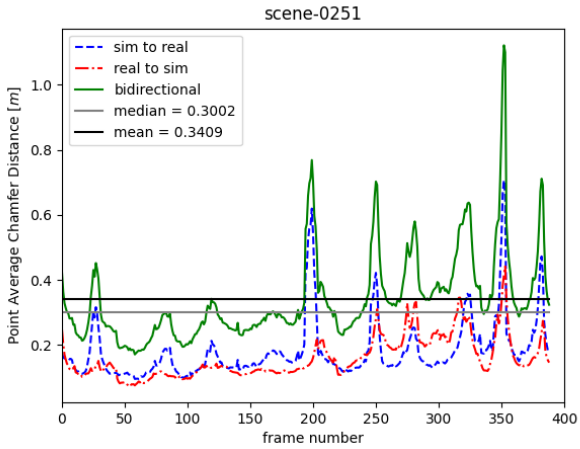
(a) scene-0131. Around frame 300, there is a speed bump in the real data, but these dynamics are not included in the simulation, resulting in a different ring pattern on the ground.



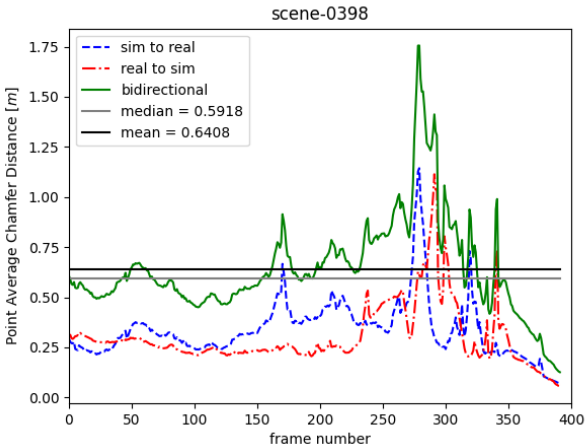
(b) scene-0132. A wrongfully cast shadow causes the peak around frame 85. That, in turn, is caused by what seems to be cars that are misplaced too far forward.



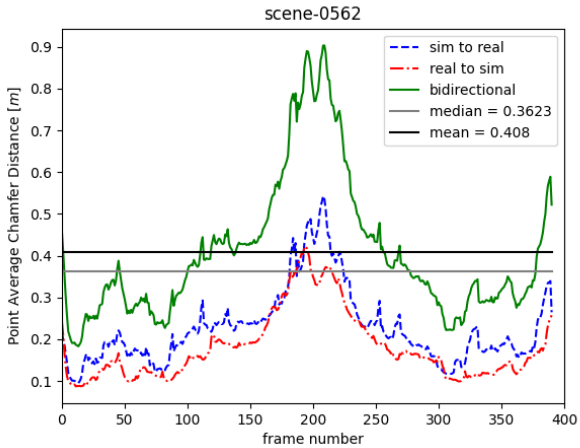
(c) scene-0164. Two times the ego vehicle has a wrong entry in the trajectory, causing the vehicle to spoof to a different location, resulting in very different point clouds.



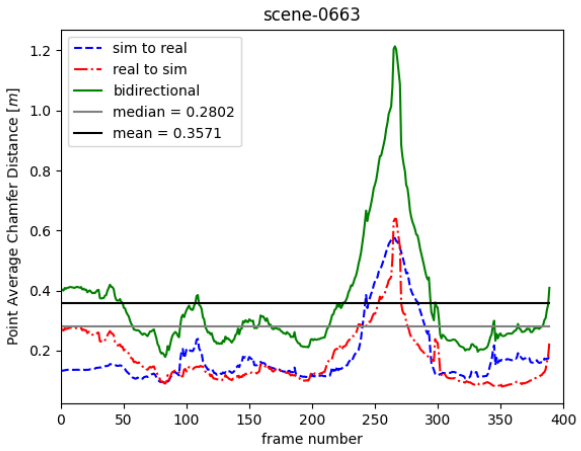
(d) scene-0251. The many smaller peaks are caused by cars passing by that cause small, miscast shadows. Some artefacts in the middle of the road, together with a larger miss-cast shadow, are the cause of the big spike around frame 350.



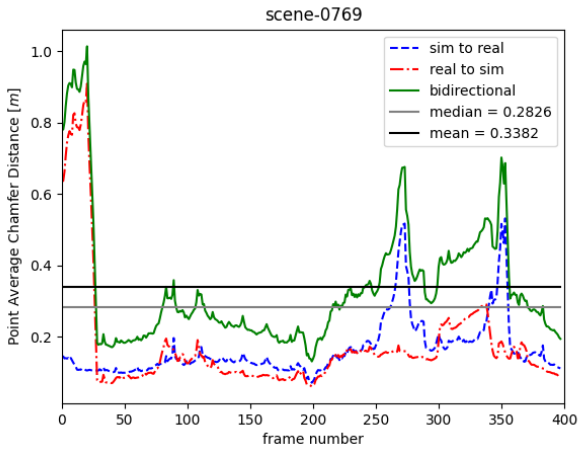
(e) scene-0398. Smaller miscast shadows cause the smaller spikes, but a larger miscast is responsible for the spike around frame 285.



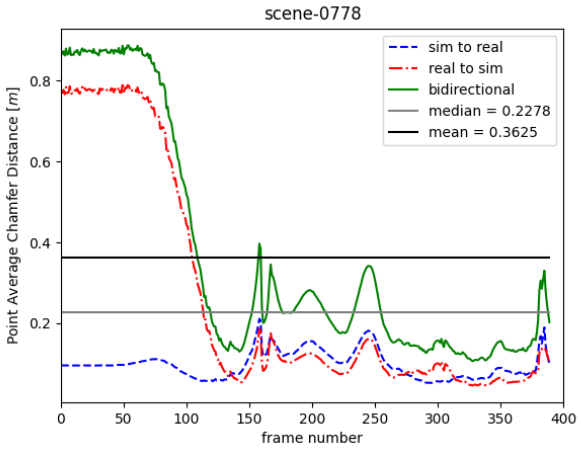
(f) scene-0562. More miscast shadows are causing spikes. the larger peak is caused by a passing car that does not seem to have the right dimensions.



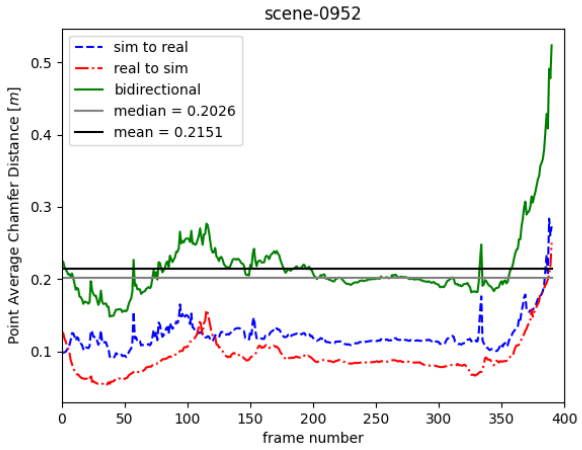
(g) scene-0663. A small truck is replaced by a too-large truck that is sticking out with its nose over the road, causing the spike around 265.



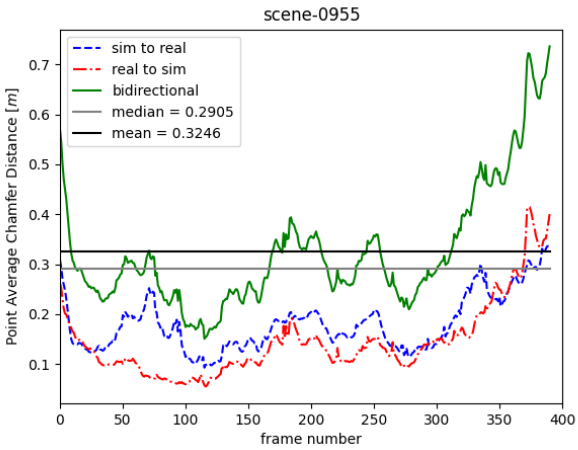
(h) scene-0796. Missized or wrongfully placed vehicles seem to be the cause of the spike.



(i) scene-0778. Until frame 100, a semi-transparent wall is non-see-through in simulation. In the real data, many lidar points exist behind the semi-transparent wall.



(j) scene-0952. Small areas of mistake in the scenery cause fluctuations, and almost no vehicles in this simulation. Nevertheless, this scene has one of the best scores.



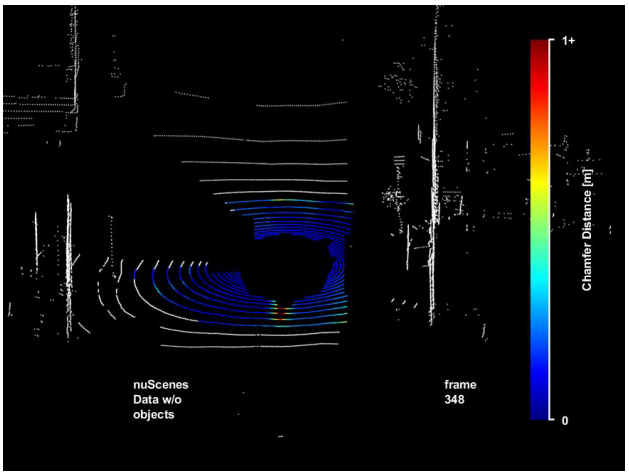
(k) scene-0955. Both a shadow miscast and two sceneries that do not seem to agree cause a rise in CD.

Figure 7. The per-frame average Chamfer Distance value of all the experiments performed in this research. Note the different y-axis scales for the different scenes. Each frame corresponds to 1/20th of a second. A short analysis of the cause of the largest peak(s) is included in the description.

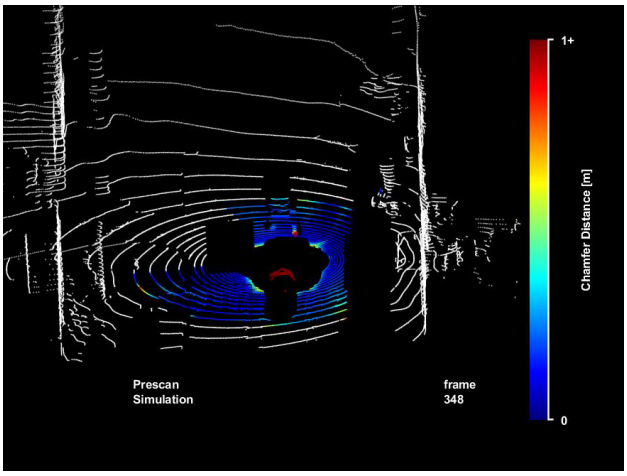
(a) The original nuScenes data.

(b) The Prescan synthetic data.

Figure 8. Camera frame 200 of scene-0251, just prior to the lidar frames shown in Figure 5. These camera images show the difference in vehicle size clearly. In the synthetic data, the artefacts are clearly visible. These artefacts are probably a result of unremoved vehicle points. The situation corresponds to that in Figure 9



(a) The original nuScenes data.



(b) The Prescan synthetic data.

Figure 9. An example of the background-only CD comparison between two frames. Only the CD of the coloured points is calculated and counted towards the average. The nuScenes data is the original data, but without objects, which have been removed using inflated ground truth bounding boxes. The simulation frame is from a simulation without (dynamic) objects, and the Ground Truth bounding box has also been removed from the scene. Of the points in the shadow of the bounding boxes, the CD is not calculated, because those points do not appear in the nuScenes frame, which in turn would cause a high CD value in simulation points that lie in the shadow of a ground truth bounding box. Note that the colour bar runs from zero to one, to emphasise which parts of the background are not performing well. The situation corresponds to that in Figure 8