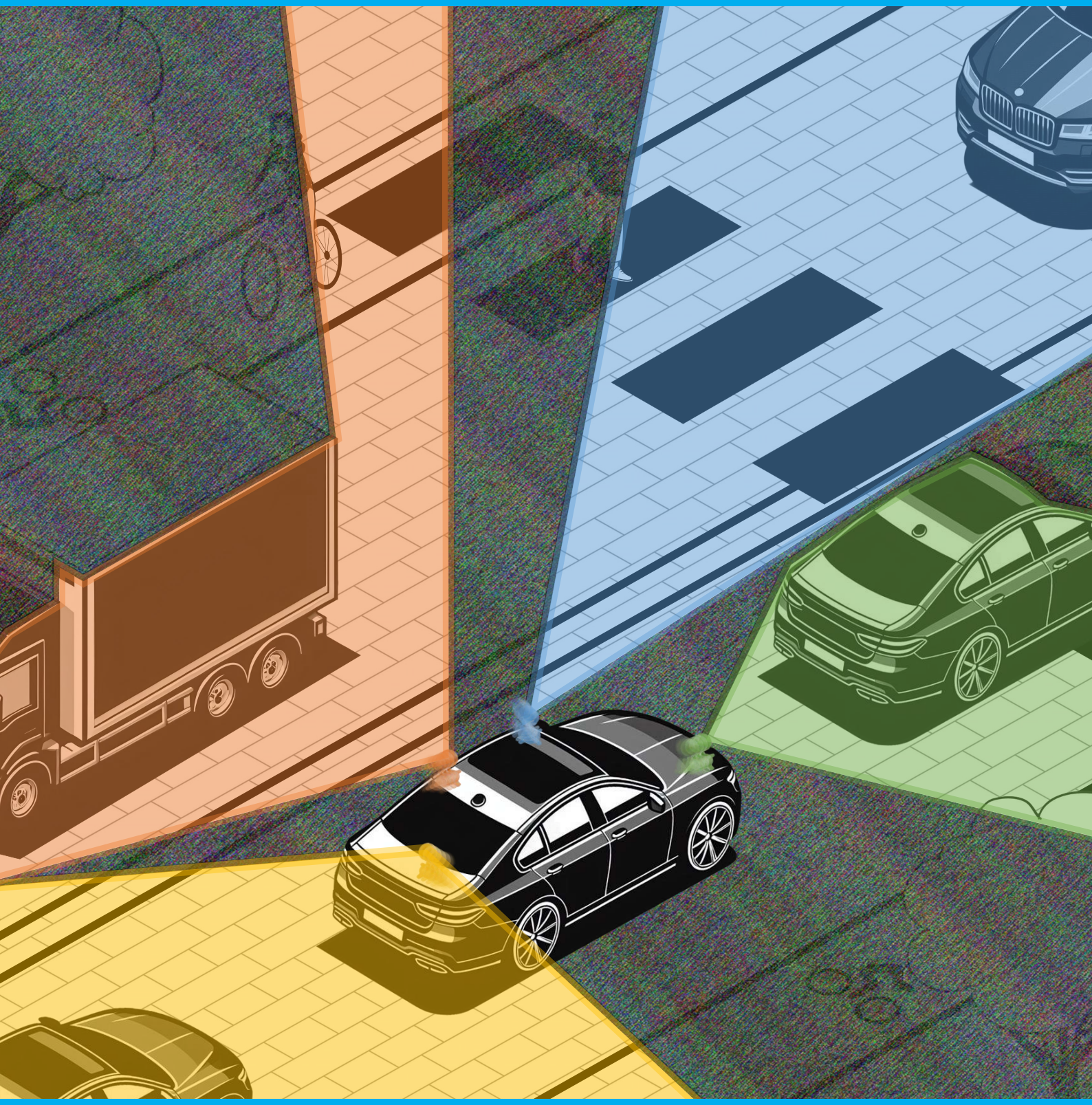
Autonomous Vehicle Sensor Placement Optimisation Using Differentiable Rendering

Master Thesis

Remco Huijsen



TU Delft

Autonomous Vehicle Sensor Placement Optimisation Using Differentiable Rendering

Master Thesis

by

Remco Huijsen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday April 16, 2025 at 12:00.

Student number:	5650844	
Project duration:	December 4, 2023 – April 16, 2025	
Thesis Committee:	Dr. H. Caesar,	TU Delft, primary supervisor
	Dr. M. Weinmann,	TU Delft, secondary supervisor
	Dr. N. J. Myers,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Autonomous Vehicle Sensor Placement Optimisation Using Differentiable Rendering

Remco Huijsen

***Abstract*—When optimising the placement of sensors on an Autonomous Vehicle (AV), research often uses evolutionary algorithms, offering a flexible way to explore complex solution spaces with multiple candidate configurations. However, this approach limits the ability to optimise one particular configuration directly. Sensor placement optimisation methods generally aim to optimise sensor-related aspects, such as visibility or coverage. When sensor placement is optimised based on the performance of a downstream task, this task performance is generally implicit, using surrogate metrics such as important task elements or probabilistic-related metrics. To address this, we propose to use gradient-descent-based optimisation in combination with a differentiable renderer. Making rendering differentiable allows gradients to flow between the initial rendering settings and the output rendered image, enabling gradient-based optimisation methods to optimise parameters based on the gradients of the objective function. This combination allows us to more directly optimise the placement of sensors on an associated downstream task. For this approach, we optimise the position of cameras on an AV based on 2D object detection performance. We create triangle mesh representations of traffic scenes from an AV simulator to use with the differentiable renderer. We use an objective function that combines optimisation losses related to in-frame rotational differences and visibility based on the detected object area of non-ego vehicles. This optimisation also considers constraints such as a minimum distance between each camera pair and camera positions that do not extend across the positioning bounds. The objective formulation results in an improved mean Average Precision (mAP) compared to a random sampling strategy and an intuitive baseline represented by cameras around the highest point on the ego vehicle.**

***Index Terms*—Autonomous Vehicles, Sensor Placement Optimisation, Differentiable Rendering, Gradient-Descent-Based Optimisation, Traffic Scene Representations**

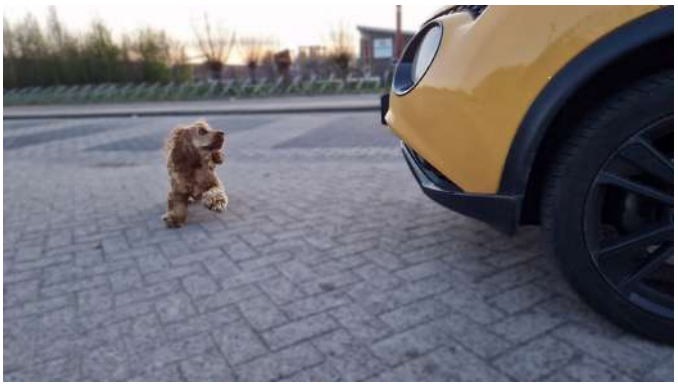


Fig. 1. Smaller objects such as children and animals require considerations regarding vehicle blind spots, as they may easily become occluded by the vehicle.

I. Introduction

Sensors are essential for allowing Autonomous Vehicles (AVs) to respond to the world around them. Camera, radar and LiDAR sensors typically ensure this environmental perception, combining different sensor data to perceive the important elements of an environment [1]. However, defining an optimal sensor configuration is challenging due to the context-dependent needs of an AV. To illustrate this context-dependence, we compare a driverless taxi and a transport vehicle in a port environment. The robot taxi drives through urban environments with a diverse selection of road users and dense groups of pedestrians. This higher density of moving objects increases unpredictability, increasing the challenge of determining which objects are high-priority. An AV should prioritise actors such as children or animals, as they are prone to behave unexpectedly. In contrast, parked vehicles on the other side of the road do not require a high priority. Additionally, smaller objects such as children and animals require considerations regarding vehicle blind spots, as illustrated in Figure 1. In contrast, a transport vehicle in a port encounters substantially fewer road users and pedestrians. This results in a more predictable environment than the urban streets. However, non-standard scenarios, such as pallets obstructing the road, can still arise, as illustrated in Figure 2. Such lower objects may be challenging to perceive consistently. This port environment also introduces environment-specific tasks that may not be necessary for urban settings. For instance, an AV in a port environment should account for overhead cranes and suspended containers. In contrast, the area above an autonomous taxi is rarely relevant for road navigation. Therefore, determining a sensor configuration for an AV requires a clear understanding of the AV-related tasks and the environment in which the AV operates.

Due to the task-dependent sensor configuration necessities, determining a sensor configuration for an AV relies predominantly on knowledge and input from senior experts. Sensor simulation software verifies that the design choices result in the desired outcome. Designing a sensor configuration involves a complex trade-off that could involve cost, sensor weaknesses, occlusions, coverage redundancy, spatial mapping, computational power, weather conditions and misalignment. The formerly mentioned sensor simulation software offers valuable insights regarding coverage and occlusions. However, translating sensor positioning to downstream task performance is less straightforward, requiring quantifying performance from various related tasks.

While environmental visibility remains important for downstream task performance, perceiving the entire environment with all actors around the vehicle does not directly dictate task performance. Research comparing various homogeneous sensor configurations for a downstream task shows increased performance solely based on the position of sensors in the configuration [2]. To illustrate the impact of sensor positioning, consider training an object detection model using dash-cam footage from behind the front windscreen. The camera position used during the data collection stage directly impacts the detectability of objects from certain classes. Assume we would now like to move the sensor to another position, for instance, to the highest point on the roof of the AV. Using the initially trained model on this new location likely negatively affects the detection performance. While the camera may still perceive objects in the environment, the roof position likely results in differing object views compared to the training set. This results in a discrepancy between the training data and the data during inference. Therefore, given an arbitrary object detection model for a particular sensor configuration, considering sensor placement requires considering object detection performance.

Fig. 2. Low objects like pallets on the road may prove challenging to perceive consistently.

Given the broad and diverse considerations for a sensor placement configuration, research proposes various sensor placement optimisation strategies to address some of the correlated challenges. Evolutionary algorithms [3]–[6] are often used to optimise the associated objective function. The majority of the related research optimises on coverage or visibility-related objective functions [4], [6]–[11]. When research considers sensor placement optimisation on downstream task performance [3], [5], strategies often use objective functions that optimise task-associated aspects or hypothetical task performance. The surrogate nature of such metrics risks the abstraction of elements related to the downstream task. Incorporating a differentiable renderer into the sensor placement optimisation procedure may provide the ability to optimise less straightforward objective functions, fueling the prospect for more direct downstream task optimisation.

The rendering procedure converts a three-dimensional object or scene into a two-dimensional image. Making rendering differentiable allows gradients to flow from the initial rendering settings to the output rendered image. The gradient represents the computational derivative of an objective function with respect to optimisable parameters. Combining this with gradient-based optimisation methods and a particular objective function allows the optimisation of all parameters based on the gradients of this objective function. Additionally, since the scene modelling in the renderer is differentiable, this increases optimisation robustness. While promising, this proposed approach presents some challenges. For instance, the objective function needs a continuous and differentiable representation, as small parameter changes should affect the objective function to provide computational derivatives. Using a differentiable renderer also requires optimisation-relevant aspects to be represented in the rendering procedure. Finally, translating the result of continuous optimisation to a real-world scenario also facilitates the need to ensure real-world constraints.

With this work, we aim to optimise the placement of sensors on an AV for the downstream task of 2D object detection, i.e. identifying and localising objects within 2D images. Using a differentiable renderer [12] allows us to optimise based on the gradient flow from the initial camera parameters to the final rendered images. Our approach addresses the research gap of more direct sensor placement optimisation on downstream task performance by incorporating 2D bounding boxes in the loss formulation. We use a pre-trained model for 2D object detection [13] to ensure good out-of-the-box performance without requiring training. Our optimisation procedure uses triangle mesh representations of traffic scenarios, replicating a traffic scene defined in an AV simulator [14]. To our knowledge, this is the first work to explore the use of differentiable rendering for AV sensor placement optimisation on 2D object detection performance. With our approach, we provide the following major contributions:

- We propose a method for recreating traffic scenes usable in a differentiable renderer. We create image sets of the environment to create point clouds, which we afterwards convert to triangle meshes. Additionally, this method allows for rendering each non-ego vehicle separately from the collective traffic scene.
- We provide multiple loss functions and show how these losses, combined with a differentiable renderer, can optimise a set of cameras on the surface of an AV. We emphasise a loss that more directly optimises camera positions based on 2D object detection performance by optimising bounding box visibility.
- Our experiments show that more explicitly accounting for downstream task performance increases downstream task-related metrics. We compare our approach to a random sampling strategy and an intuitive baseline of cameras around the highest point on the ego vehicle.

II. Related work

In this section, we discuss research related to differentiable rendering and sensor placement optimisation. For sensor

placement optimisation, we distinguish between research that explicitly optimises for downstream task performance and research that does not do this.

Differentiable Rendering. This research addresses differentiable rendering to optimise the position of cameras on an Autonomous vehicle (AV). The two most common rendering techniques are rasterisation and ray tracing. The main difference between these methods is that rasterisation is an object-centric approach and that ray tracing is an image-centric approach. Rasterisation is object-centric because it uses perspective projection to convert a 3D triangle representation into a 2D screen representation [15]. Ray tracing is an image-centric approach because it traces a light ray's path through the centre of each pixel into the three-dimensional scene, checking if it intersects with an object in this scene. Ray tracing generally results in more realistic-looking images since it accurately represents lighting phenomena, at the cost of being more computationally expensive.

Both rasterisation and ray tracing do not explicitly account for differentiability. The literature proposes several different methods to enable differentiable rendering. For rasterisation, for instance, research proposes to use a soft rasterisation approach that uses a blur radius and the blending of faces to address rendering discontinuities [16], use an approximate gradient to render polygon meshes from two-dimensional images using neural networks [17], or compute gradients analytically by separately rasterising the foreground and background with a weighted interpolation of local properties and a distance-based aggregation of global geometry respectively [18]. A related method is surface splatting, a rasterisation technique for point clouds, made differentiable by a stochastic representation that models the contribution of a point as a probability distribution [19]. For ray tracing, methods address discontinuities by using reparametrisation techniques [20] or with the Reynolds transport theorem and boundary integrals [21].

Differentiable rendering has enabled significant advancements for implicit representations, notably with NeRF [22] and 3D Gaussian Splatting [23]. Other tasks that benefit from differentiable rendering are tasks such as camera pose estimation [24]–[28], object pose estimation [29]–[31], and optical parameter estimation [32]. The versatility of optimisation problems using differentiable rendering techniques in the literature demonstrates the potential for sensor placement optimisation with image-related objective functions. Especially the work considering camera pose and object pose estimation relates to this idea, showing that an adequate objective formulation based on the images and the gradient flow throughout the rendering procedure suffices.

Sensor Placement Optimisation. Sensor placement optimisation methods aim to optimise the position of a sensor based on the optimisation of an associated objective function. The concept of sensor placement optimisation dates back to (at least) 1973 with the art gallery problem [33], [34]. The art gallery problem asks how many guards are needed to fully cover the visible area within an art gallery, where the gallery is a simple polygon. Optimising for sensor visibility or coverage has remained a predominant approach in sensor placement optimisation strategies. This concept can be related to research optimising for area coverage [8], [11], surround-view coverage [4], [6], [10] or object visibility [7], [9]. The to-be optimised sensor or sensor configuration can be related to an existing sensor such as a (depth) camera [4], [7], LiDAR sensor [6], ultrasonic sensor [8] or a more general sensor representation [9]–[11]. Generally, sensor placement optimisation formulations get separated into discrete and continuous formulations.

A discretely formulated problem has a fixed subset of possible sensor locations. For these discrete formulations, strategies such as integer programming [9] and brute-force search [10] are regularly used to sample all possible combinations. Another approach for discrete formulations is to have a large set of sensors in a 3D space and use a random sample consensus algorithm to randomly pick a subset of sensors to optimise an objective function [7]. The main downsides of using discrete methods are the computational expenses needed to sample each possible combination of sensor positions. Additionally, the fixed solution subset could result in the optimal position not being represented. While discrete formulations allow for more straightforward objective functions, such as binary visibility for objects, one can argue about their usefulness.

In contrast to discrete problem formulations, continuous formulations are not limited to a fixed number of camera poses. Continuous formulations allow sensor positioning anywhere within a specified region. For continuously formulated problems, evolutionary algorithms, such as Particle Swarm Optimisation (PSO) [4], the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11], and the Genetic Algorithm (GA) [6], are regularly used to optimise the placement of sensors. Evolutionary algorithms are population-based heuristic-based methods usable for both continuous and discrete problem formulations. These algorithms operate by iteratively evolving candidate solutions through evolutionary-based computations to ensure fit solutions. Gradient-based optimisation methods [8], [9], [11], [33], optimising based on the gradients related to an objective function, are also well represented in the literature. The gradient represents the computational derivative of the objective function regarding the optimisable parameters.

Which optimisation strategy is preferable depends on the problem and the objective formulation. For less complex problems, optimisation methods such as integer programming or a brute-force search may be preferable in contrast to gradient-based optimisation methods or evolutionary algorithms. However, as the complexity of the problem grows, so will the search space, making these optimisation methods computationally expensive. While gradient-based optimisation methods and evolutionary algorithms can mitigate this issue, they also have associated challenges. For instance, one must account for these optimisation methods getting stuck on local minima rather than the global optimum. We only encountered one research that used gradient-based sensor placement optimisation combined with a differentiable renderer [9], [12]. However, this work focuses on roadside sensor configuration

on rails above the road. Whilst this research illustrates the potential of combining differentiable rendering with sensor placement optimisation, the roadside sensing context does not fully align with our AV context. The higher-up sensor position along vehicle lanes inherently has a lot of coverage of the environment and the actors within it. In contrast, sensors on an AV have a more limited sensing range due to the lower positioning, and this position results in many occlusions from the surrounding road users and pedestrians.

Sensor Placement Optimisation For Downstream Task Optimisation. A subset of sensor placement optimisation methods consider the performance of a particular downstream task as their objective function. Research optimises for downstream tasks such as tracking and detecting non-ego vehicles for camera and radar configurations [5], and 3D object detection and semantic segmentation for LiDAR configurations [3]. For this, they employ strategies related to the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [3], Simulated Annealing with Greedy Random Adaptive Search Procedure (SA+GRASP) [5], the Genetic Algorithm (GA) [5], and Particle Swarm Optimisation (PSO) [5]. Related research also proposes surrogate metrics where the optimisation of the proposed metric correlates with the optimisation of a downstream task, for instance, metrics that estimate the information gain for occupying regions in a probabilistic occupancy grid [2], [35] and entropy-based metrics [36], [37]. Methods have correlated the result of their optimisations to performance on the downstream task and noted that optimising with their method and/or metric generally increases the performance on the downstream task. For instance, research proposing a data-driven surrogate metric with maximum information gain for the object detection performance of LiDAR sensor configurations noted that positioning contributes up to 10% performance discrepancy regarding average precision in challenging environments. However, the surrogate nature of the used metrics generally results in implied downstream task optimisation. For instance, research proposes to optimise based on a weighted sum of configuration characteristics to track and detect non-ego vehicles [5]. While these aspects may be crucial for the downstream task, this reconstruction, as the sum of important aspects, likely fails to consider full task performance. Even when using a more probabilistic-based surrogate metric, such as with probabilistic occupancy grids [2], [35], semantic occupancy grids [3], and entropy-based metrics [36], [37], the optimisation is grounded in hypothetical downstream task performance. Therefore, we believe there is a gap in the literature regarding more direct optimisation on a particular downstream task. We address this literature gap by more directly optimising sensor positions on downstream task performance, specifically by optimising cameras on the downstream task op 2D object detection. Leveraging a differentiable renderer, with gradient flow between the initial rendering settings and the output rendered image, allows us to optimise based on the gradients of an objective function. Relating this objective function towards the downstream task would allow this more direct optimisation.

III. Methodology

This section goes over our proposed methodology to:

- Optimise the placement of cameras on an AV [14].
- Optimise based on the downstream task of 2D object detection.
- Optimise by utilising a differentiable renderer with gradient-descent-based optimisation.

The methodology section is segmented into three sections, as illustrated by Figure 3.

Section A focuses on representing a traffic scene from an Autonomous Vehicle (AV) simulator as a triangle mesh. Using a differentiable renderer requires representing optimisation-relevant aspects. In our case, this relates to the representation of a traffic scene with an ego vehicle and non-ego vehicles to detect. The use of a pre-trained 2D object detection model [13] necessitates a relatively realistic coloured scene representation to detect objects in the environment consistently. However, this triangle mesh representation mainly supports the sensor placement optimisation procedure. Therefore, we prioritise creating ideal testing data over utilising real-world data. To this end, we aim to set up a traffic scenario in an AV simulator [14] and convert this to a triangle mesh representation usable in a differentiable renderer.

Section B focuses on prerequisites to enable optimisation. These prerequisites serve to allow the gradient-descent-based optimisation procedure. This section focuses on camera positioning on the surface, the images used to determine optimisation-relevant aspects, and how GT bounding boxes are created and used in the optimisation.

Finally, Section C describes the gradient-descent-based optimisation procedure. One difficult aspect concerning continuous optimisation is enforcing constraints. We propose four losses, two constraint losses and two optimisable losses. Our constrained optimisation procedure aims to ensure satisfied constraints whilst optimising for the optimisable losses. We also allow direct scaling of the loss values, increasing the numerical importance or the associated gradient magnitude. Given a singular optimisable loss L_{o} and its associated scalar value S_{o}, the number of constraint losses n, the associated Lagrange multiplier λ_i and constraint loss value L_{c_i} with its associated scalar S_{c_i}, the constrained optimisable loss L_{co} is determined as in Equation 1. The total loss L is the sum of m optimisable losses, as also indicated in Equation 1.

$$\begin{aligned} L_{\mathrm{co}} &= S_{\mathrm{o}} \cdot L_{\mathrm{o}} + \sum_{i=1}^{n} \lambda_i \cdot S_{\mathrm{c}_i} \cdot L_{\mathrm{c}_i} \\ \text{Minimise: } L &= \sum_{j=1}^{m} L_{\mathrm{co}_j} \end{aligned} \tag{1}$$

A. Traffic Scene Representation

This section describes the process of representing and capturing images from the environment inside an AV simulator

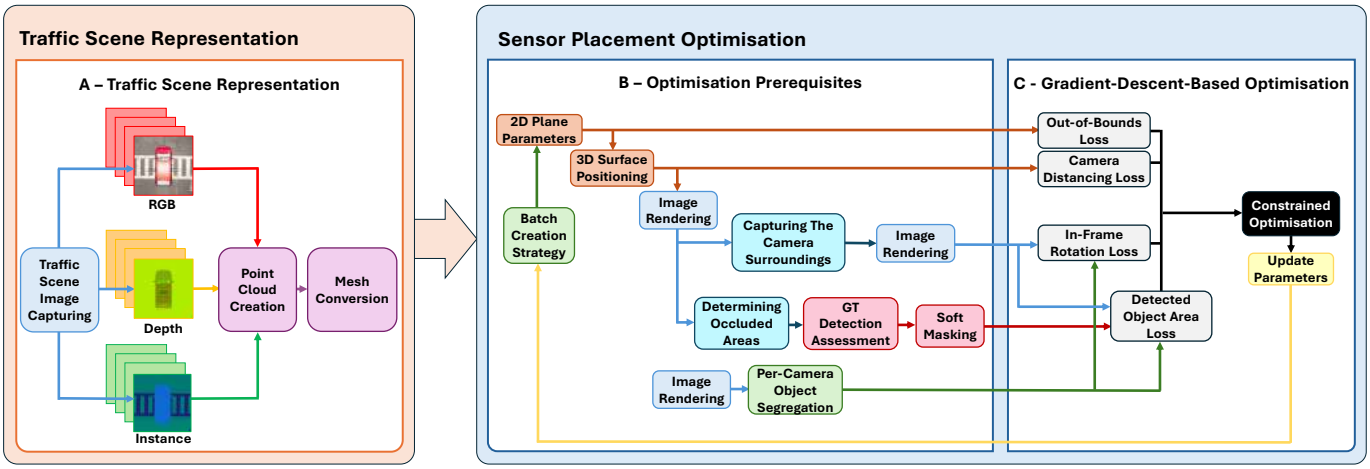


Fig. 3. The figure shows an overview of our AV sensor placement optimisation method for optimising cameras on 2D object detection performance. We segment the methodology into two parts: representing traffic scenes on the left and sensor placement optimisation on the right. We divide the sensor placement optimisation procedure into optimisation prerequisites and gradient-descent-based optimisation.

[14], using these images to create aggregated point clouds, and converting these point clouds into separate triangle meshes.

Traffic Scene Image Capturing. The first step in our process is to set up the traffic scenario within the AV simulator [14] and to capture multiple image sets during and after this process to reconstruct this scenario effectively. For each 6 Degrees-Of-Freedom (6DOF) camera pose, we capture a set of three distinct image types:

- A RGB image providing the colour per pixel.
- A depth image providing the absolute depth per pixel.
- An instance image providing what pixel correlates to which uniquely spawned object.

Figure 4 illustrates these three image types for one identical camera pose. We mitigate lighting and weather effects to ensure environmental consistency from any camera position in the scene. Since we are taking the images subsequentially, simulating aspects like rain and wind will make it challenging to represent a cohesive environment. We mitigate lighting effects such that reflections, lens flares and shadows do not result in noise. This noise can occur when two images from different camera positions represent an identical part of the traffic scene differently. Lighting can, for instance, result in illumination changes, lens flares and slightly different colours due to how the light from a particular angle lights and reflects. We also try to minimise the effects of harsh shadows so that we know that each object in the scene should be detectable. We position the sun overhead to mitigate these harsh shadows, minimising shadows on vehicle surfaces. Additionally, we remove vegetation to ensure that we can effectively capture each aspect of the environment without needing to account for obstruction from objects, like trees that obstruct top-down images. Starting from (1) in Figure 5, presenting a point cloud of an unaltered simulator environment, we first remove the vegetation as in (2), whereafter we alter the lighting and weather parameters as in (3).

Setting up vehicle scenarios entails generating valid spawn points and populating the scene with a consistent ego vehicle and a set of randomly coloured non-ego vehicles within a spawn radius relative to the ego vehicle. We define the set of valid spawn points as the collection of set-up spawn points in the simulator, supplemented with points based on the road network. This collection of spawn points ensures that every possible spawn location relates to normal driving behaviour. We capture instance images before and after spawning each vehicle to ensure that the non-ego vehicle is visible from the ego vehicle's position. We assess visibility based on whether the second instance image has an additional vehicle instance in the image frame. We capture various image sets from different perspectives, including above the (non-)ego vehicles, top-down at multiple heights, and multiple rotational steps from the ego vehicle's location. We explicitly capture the roof of the ego-vehicle to represent the valid sensor positioning area later. To better understand all used images to recreate one traffic scene representation, we refer to Appendix VIII. For non-ego vehicles, capturing the roof reduces the number of non-captured aspects of the vehicle, reducing artefacts during the mesh conversion. Note that this means we do not fully capture all aspects of the traffic scene but focus on the aspects visible from the ego vehicle. For each captured image, we also store the 6DOF camera pose. This 6DOF pose is relative to the position and orientation of the ego vehicle. This relative position ensures that each traffic scene representation is structured similarly. Besides our collection of image sets with their associated 6DOF camera poses, we also save additional information about the spawned vehicles, like the spawn points, the surrounding 3D bounding box dimensions, and all the associated instance colours. The addition of non-ego vehicles in the traffic scene representation is visible in (4) of Figure 5.

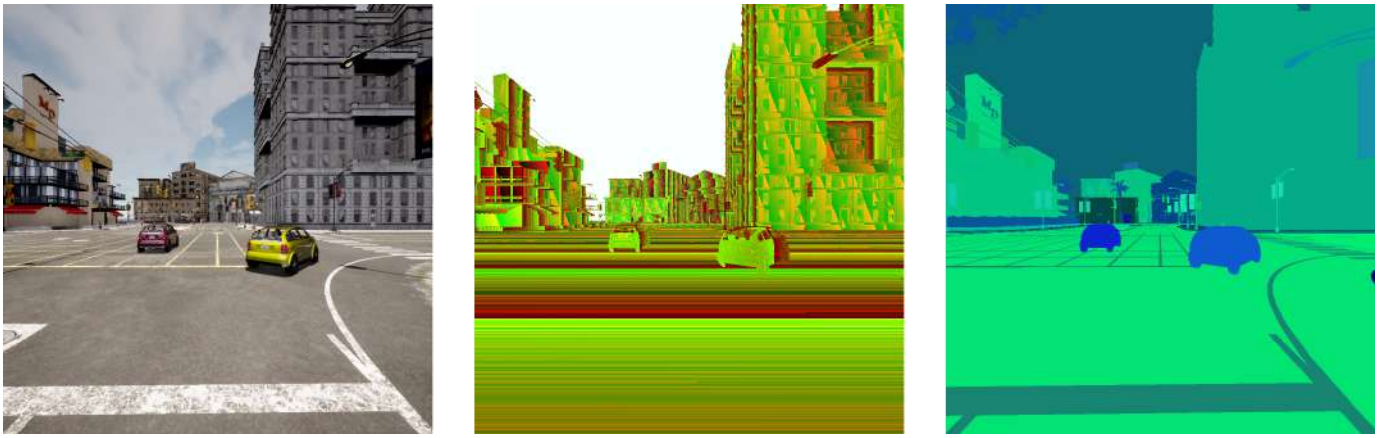


Fig. 4. The figure shows three image types from the same camera position, captured in an AV simulator [14]. The left image shows the RGB image providing the colour per pixel. The middle image shows the depth image, encoding the absolute depth [38]. The right image shows the instance image, encoding unique object instances.

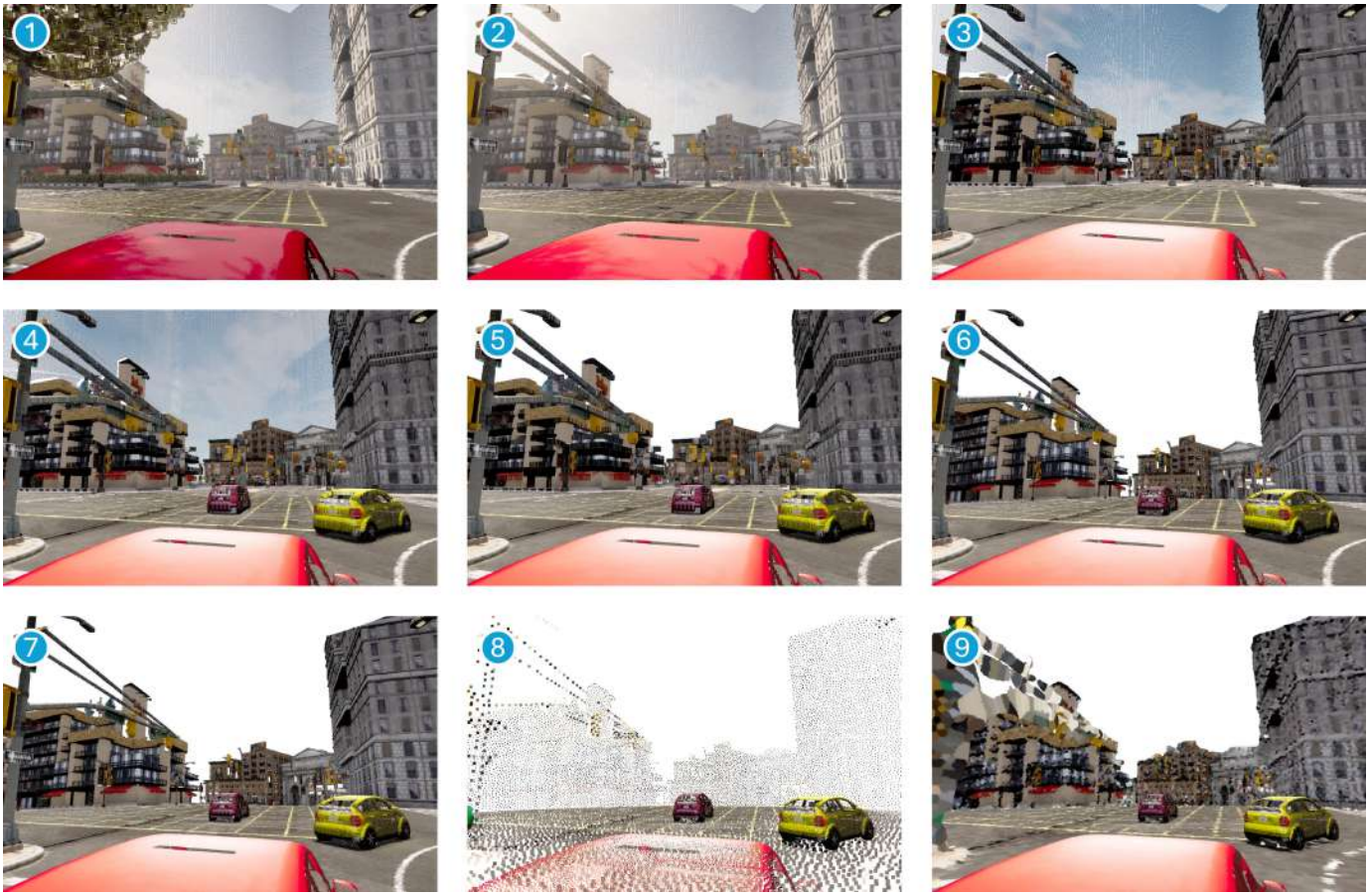


Fig. 5. The figure illustrates different representations of a traffic scene initially set up in an AV simulator [14]. (1) shows a point cloud representation of an unaltered traffic scene. The next images represent the following sequence of global operations: (2) Remove all vegetation; (3) Minimise weather effects and move sun overhead; (4) Add non-ego vehicles; (5) Filter far-away points; (6) Convert points outside of the representation radius into a sphere representation; (7) Filter points above a given height; (8) Downsample each point cloud; (9) Convert each downsampled point cloud into a triangle mesh with NKSR [39].

Point Cloud Creation. We create aggregated point clouds based on our captured images and their associated 6DOF camera poses. Given a depth image and its associated camera pose, Field Of View (FOV) and image resolution, we can convert these absolute distances from the camera into a point cloud in 3D space. Combining this with the correlated RGB image creates a coloured point cloud. Finally, the instance images allow the separation of points for each instance of interest. Here, we distinguish between two point cloud types: individual point clouds of the (non-)ego vehicle(s) and a point cloud representation of the environment around these actors. However, creating an aggregated point cloud of the environment by simply reprojecting points would result in an overly large point cloud capturing everything in sight, including large buildings and the sky. Therefore, we mitigate the scale of point clouds twofold. Firstly, we filter out depths in the depth images that approach the upper depth threshold. These values mainly relate to points corresponding to the sky, as illustrated with (5) in Figure 5. Secondly, we account for the absolute distance between all points and the spawn point of the ego vehicle. For all points with an absolute distance larger than the desired representation radius, we convert these points to spherical coordinates, adjust the radius to the desired representation radius, and transform the point back to cartesian coordinates. This transformation effectively creates a sphere representing the environment beyond the radius. (6) in Figure 5 shows how this environment representation looks from the ego vehicle's point-of-view. Figure 6 shows this from a top-down view, illustrating how a large environment, as observable on the left, is essentially represented by a compact sphere, as observable on the right. This representation gives the illusion of an environment while limiting the actual environment space. We perform additional filtering to filter out points above a maximum height, as this predominantly relates to larger structures that are unimportant for our objective, as illustrated with (7) in Figure 5.

One important consideration when using RGB images, depth images, and instance images to create an aggregated point cloud is conflicting information. Suppose we have an RGB image, a depth image and an instance image of a car in a traffic scene. In the RGB image, we may still perceive parts of the environment through the vehicle's windows. Our depth image also gives the correct depth values for the environmental presence through the windows. However, if we take an instance image of the vehicle, we lose the translucency from the windows because the windows are part of the vehicle instance. Therefore, if we solely rely on these three images, the environment through the windows will be seen as part of the vehicle instance. To combat this, we rely on the vehicle's 3D bounding box. Given this 3D bounding box, we state that only points inside the 3D bounding box count towards belonging to the vehicle, and only points outside the bounding box count toward belonging to the environment. Figure 7 illustrates the effect of this procedure, with many environmental artefacts in the left image and these parts removed in the right image.

The ego vehicle's point cloud is unique, as it doubles as the valid camera positioning area. Given a point cloud of the ego vehicle, we estimate the normals and filter for normals with an approximate upward direction. This filtering procedure leaves us with the roof and hood of our ego vehicle. We still use both ego vehicle point clouds: the filtered surface point cloud for camera placement positioning and the unfiltered surface point cloud for visualising the ego vehicle.

Mesh Conversion. The final step in creating our traffic scene representation is the conversion of point clouds to triangle meshes. The main benefit of a triangle mesh representation is that, compared to points in space, it has a structure which connects these points. Since our aggregation strategy results in a dense point cloud with some noise, this requires a fitting mesh conversion method. Due to the point density, downscaling the number of points is likely needed. However, this likely does hamper the use of certain surface reconstruction methods. For instance, using a Poisson surface reconstruction [40] with a sparse or noisy point cloud likely results in overly smoothed objects and mirrored normals, mitigating the effectiveness of the representation. To mitigate these problems, we use a Neural Kernel Surface Reconstruction (NKSR) method [39]. In short, NKSR is a surface reconstruction algorithm using the Neural Kernel Field (NKF) representation [41], allowing the reconstruction of 3D implicit surfaces from large-scale and noisy point clouds. This surface reconstruction strategy is a good fit, given our point cloud aggregation strategy. This work explicitly addresses high-end reconstruction results for AV driving scenarios, both from an AV simulation [14] and an open data set [42], strengthening the choice for this method. An additional benefit of this method is that it allows the estimation of normals based on the original positions of the sensors that perceived specific points, which we know from our point cloud aggregation strategy. This position-based normal estimation method results in consistently better mesh normals, substantially mitigating mirrored normals. Before converting the point clouds to a mesh representation, we downsample the point cloud to a desired number of points using a Poisson Disk strategy [43]. We chose the Poisson Disk strategy to directly downsample the points to approximately a specified number of points, ensuring adequate point spacing to maintain the point cloud's original shape. While this strategy likely results in lessened visual fidelity, we mitigate the impact on object detection performance by predominantly focusing on downsampling the environment rather than the non-ego vehicles. (8) in Figure 5 shows the extend of the downsampling. After the mesh conversion, we perform additional filtering to reduce the complexity of the mesh. We first delete duplicate vertices and faces and afterwards use a quadric-based edge-collapse strategy [43], [44] to downsample to a specific number of faces. The choice for the quadric-based edge-collapse strategy is similar to the Poisson disk strategy, such that we can directly downsample the faces to approximately a specified number of faces whilst ensuring that the faces still represent the original mesh. Both the point cloud and the mesh downsampling reduce the computational expenses for the mesh conversion and the

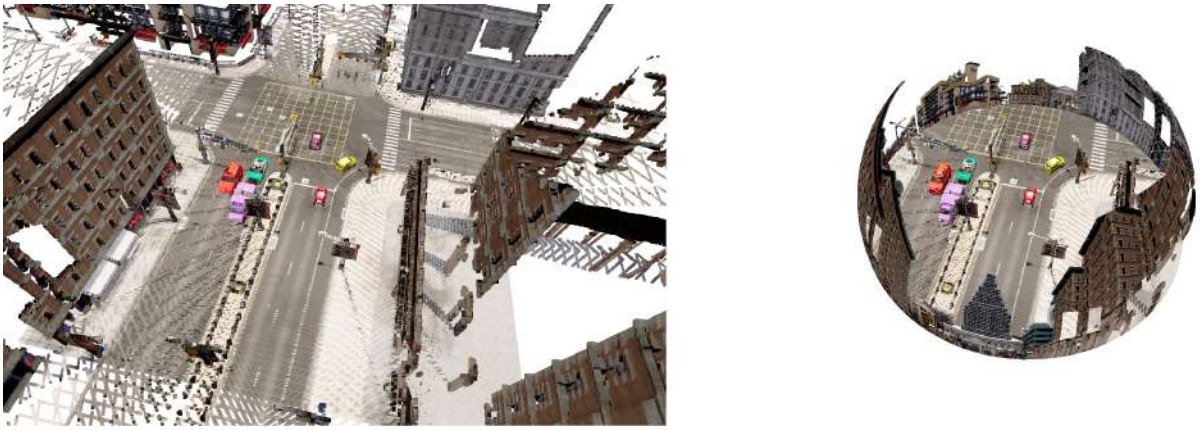


Fig. 6. The figure illustrates the usefulness of the spherical environment representation. The left image shows the result of not accounting for the distance of points from the ego vehicle, creating a large-scale point cloud representation. The right image shows the result of our spherical representation, where a sphere around the environment represents the environment beyond the representation radius. This spherical representation mitigates the size of the point cloud whilst keeping elements outside of the representation radius part of the traffic scene.

Fig. 7. The figure illustrates the need to account for points visible through vehicle windows. The left image shows the vehicles with many environmental artefacts around them, now considered part of the vehicles. Only associating points within the 3D bounding boxes of particular vehicles results in the right image, with most artefacts removed.

representation in the differentiable renderer. (9) in Figure 5 shows how the triangle mesh representation of a particular traffic scene in our approach looks.

Traffic Scenes Collection. With our proposed traffic scene representation methodology, we created around 200 traffic scene representations. This number of scenes allows us to divide this scene collection into a training and test set while maintaining a varied selection of distinct traffic scenarios. We only use the training set to train the model. The test set is not part of the training, allowing an unbiased evaluation result. To illustrate the occupancy of vehicles relative to the ego vehicle, we created top-down heat maps of the vehicles in all environments. Figure 8 shows the result of this procedure for all traffic scenarios, the training set, and the test set. For each heat map, dark red represents the highest occupancy presence, and dark blue represents the lowest. The prominent low-occupancy area in the middle of each image is the position of the ego vehicle. Each heat map shows a comparatively higher object presence on the left side of the ego-vehicle. This occupancy discrepancy is likely due to our vehicle spawn point approach and the suburban environment we used in the AV simulator. As this environment represents a downtown area, there are limited vehicle lanes, and the pavement often occupies the right side of the vehicle. Only considering vehicles on the road logically results in the left side of the vehicle generally having a higher occupancy presence. The distinct traffic-lane-like lines in the heat map are also likely a result of these spawn points.

B. Optimisation Prerequisites

This section addresses some of the aspects required to optimise cameras on the performance of 2D object detection. Aspects entail which information from the differentiable renderer we use, how we ensure valid camera positioning, the assessment of True Positive (TP) detections, and how we use a camera to represent the rotational space.

Image Rendering. Using a differentiable renderer allows us to render images given specific parameters, like the 6DOF

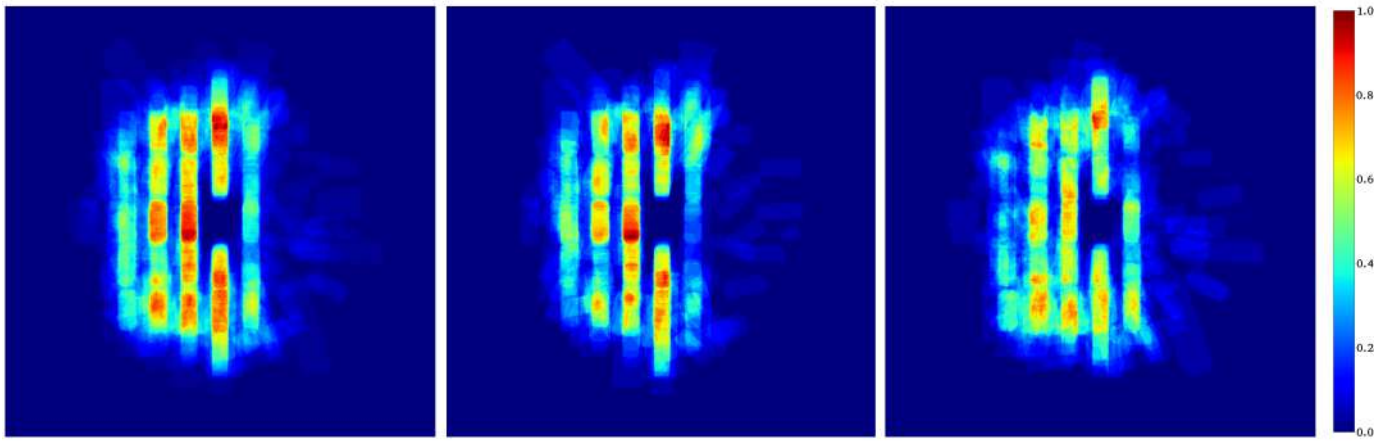


Fig. 8. The figure shows top-down heat map representations of the non-ego vehicle distributions in the traffic scene representations. The left heat map represents the occupancy of all vehicles in all traffic scenes. The middle heat map shows the distribution of vehicles in the training set. The right heat map shows the distribution of vehicles in the test set. In each heat map, dark red represents the highest occupancy, and dark blue represents the lowest occupancy. The non-occluded space in the middle of the images is the position of the ego vehicle.

pose, the Field Of View (FOV), and the resolution, whilst maintaining gradient flow. We require three image types:

- A silhouette image, essentially a mask of the rendered object(s) in the image frame.
- A RGB image, essentially a render of the textured object(s) in the image frame.
- A rasterisation z-buffer, essentially a relative depth image of the object(s) in the image frame, scaled between the closest and the furthest surface in the image.

Per-Camera Object Segregation. To mitigate the complexity of optimising multiple cameras on object detection, we implement a segregation strategy which couples each vehicle from each environment to one specific camera. This segregation strategy effectively results in the detectability of each vehicle, which only impacts the optimisation of one camera instance. This segregation strategy mitigates the need to account for inter-camera coordination. For this, we position the camera in the middle of our traffic scene representation and create a silhouette image for each distinct vehicle in each traffic scene representation for a set number of orientations. This procedure informs us which vehicles from which traffic scenes are visible in which orientations. Afterwards, for n optimisable cameras, we find the n number of orientations that maximises the number of vehicle instances visible in a camera frame. Combining orientations to maximise visible vehicle instances may leave us with multiple orientation sets that result in the same amount of visible instances. For these sets, we assess which combinations of camera orientations result in the lowest number of duplicate objects in the frame. We randomly select a set if this check still results in multiple sets. For the final set, we identify vehicles present within multiple camera orientations. To ensure each object correlates to one camera, we associate the vehicle with the orientation where the object's visibility is the highest. After this final procedure, we have a set of n segregated vehicle instance sets. We associate each vehicle set with a camera index for the optimisation procedure.

Batch Creation Strategy. We use an informed batch creation strategy to compel optimising all cameras. Since we know which vehicles from which traffic scenes correlated to which camera indices from our segregation strategy, we also know what scenes optimise which cameras. Therefore, we can construct the traffic scenes batch based on this information. If the number of cameras fits the batch size, we randomly select a camera-correlated environment for each camera index. Otherwise, we randomly select a camera index and an associated traffic scene. While this likely still results in specific cameras seeing more optimisation in general, it should provide more consistent optimisation for cameras correlated to less populated areas.

3D Surface Positioning. Recall from Section A that we have a point cloud representing the valid positioning area on the ego vehicle. We aim to convert this surface representation into a height map, where the correlated height for each 2D position is implicit. For this, we determine the minimum and maximum boundaries of the point cloud and create a raster with a set raster size. We then iterate over each raster and assess the highest point within each raster. Applying this procedure to all rasters leaves us with a height map for all rasters with associated points. However, imperfections or non-captured areas likely result in empty rasters for an arbitrary point cloud. For our point cloud specifically, we have a large empty area without points related to the windows of the ego vehicle. Therefore, we interpolate missing values by the surrounding present raster values. As this might result in an uneven representation, we iteratively smooth the interpolated values. We continue this smoothing procedure until the maximum difference between two iterations is below a certain threshold. With this, we have a smooth surface representation of where we can position cameras, with the height implied

from the 2D position. Figure 9 shows the initial point cloud representation with overlaid coloured pixels representing the height at each height map raster.

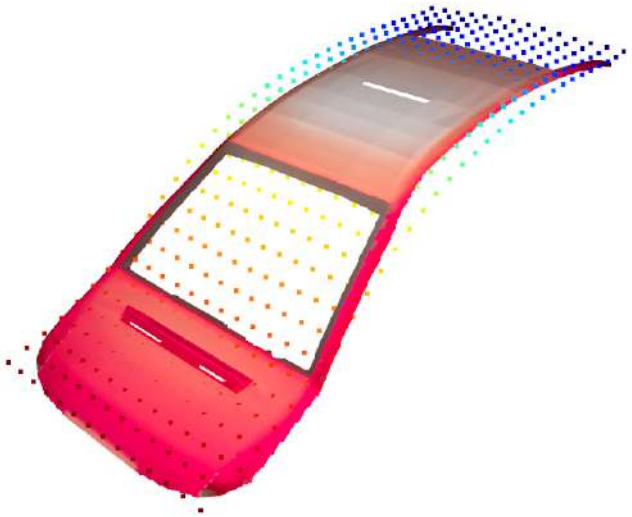


Fig. 9. The figure shows the valid positioning area of the ego vehicle, represented as a point cloud, overlaid with coloured pixels representing the height correlated to a raster within the height map.

Determining Occluded Areas. We desire to assess which parts of which vehicles are occluded, as occlusions affect the visibility and detectability of vehicles. To this end, we assess the occluded areas of a vehicle by using silhouette images and rasterisation z-buffers. We first assess which vehicles in the camera frame overlap. If we have two silhouette renders of two separate in-frame vehicles, we determine that these vehicles overlap if any part of the two silhouettes overlap. If we do this for all vehicles, we know which vehicles potentially result in occlusion. For a particular vehicle and a set of possible occluding vehicles, we create separate z-buffers for each combination of the vehicle of interest and each possibly occluding vehicle. The z-buffer per combination represents the relative depth for the two in-frame vehicles, meaning we can assess which vehicle is closer. To assess which vehicle is closer, we specifically look at the non-overlapping area for each object. If a possibly occluding vehicle is closer to the camera, this vehicle is occluding this vehicle of interest. Otherwise, the vehicle of interest is not occluded but is occluding the other vehicle. Applying this procedure for all possibly occluding vehicles allows us to assess the non-occluded area of a vehicle. Additionally, we assess how much the ego vehicle occludes the vehicles in the environment. Assuming that the ego vehicle is always the closest vehicle to the camera, only using silhouette images suffices for this.

Capturing The Camera Surroundings. One of the challenges of optimising for downstream task performance on cameras is that the optimisation is limited to what is inside the image frame. While we could use external data, such as positional data from the triangle meshes, such inclusions limit the effectiveness of our approach. Therefore, we only optimise based on data related to the image frame or the position of the cameras. However, we must consider how we handle relevant aspects that are not currently within the frame. Brilliant 2D object detection performance does not entail much when most vehicles are not perceivable. To this end, we propose to segment the rotational space of 2π based on the FOV of our camera and assess the three-dimensional space from the camera's current position. Given FOV F, we determine the closest divisible FOV F_{closest} as in Equation 2. Each rotational addition θ_a for each addition a follows from Equation 3. Combined with our traffic scene representation with separated vehicle meshes, we can perceive the environment around a given camera position, both the collective traffic scene and each distinct vehicle.

$$F_{\text{closest}} = \begin{cases} F & 2\pi \bmod F = 0, \\ \frac{2\pi}{\text{round}\left(\frac{2\pi}{F}\right)} & \text{else} \end{cases} \tag{2}$$

$$\theta_a = a \times F_{\text{closest}} \quad \text{for } a = 0, \ldots, \frac{2\pi}{F_{\text{closest}}} - 1 \tag{3}$$

Ground-Truth Detection Assessment. We must assess 2D object detection performance based on our traffic scenario representations. However, such an assessment requires knowledge of the Ground Truth (GT) bounding boxes correlating to the best possible detection performance. Relying on predetermined bounding boxes becomes challenging when the continuous optimisation can position a camera anywhere within the valid positioning region. To combat this, we propose a method to determine a GT bounding box from anywhere in the environment. To accomplish this, we rely on silhouette renders representing the non-occluded area of each distinct vehicle. Given the non-occluded silhouette, we determine the largest group of connected pixels of this silhouette and create a GT bounding box based on the minimum and maximum x and y values. When assessing detection model bounding boxes, we deem a bounding box from an object detection model a True Positive (TP) detection for a particular GT bounding box if:

- The detection bounding box confidence adheres to a minimum confidence threshold.
- The Intersection over Union (IoU) adheres to a minimum IoU threshold.
- Any part of the GT bounding box correlated object falls within the detection model bounding box.
- Neither the GT bounding box nor the detection model bounding box has previously been matched.

If multiple detection model bounding boxes comply with these criteria, we match the bounding box with the highest IoU.

Soft Masking. Masking parts of an image is an often-used strategy for computer vision-related tasks. However, the boolean nature poses a challenge in a differentiable optimisation procedure. Zero values in the mask likely disrupt gradient flow correlated to certain parts of the image. While

there are more direct methods to compute masks differentially, for instance, using a sigmoid, these masks are relatively imprecise compared to discrete operations. Research has proposed several methods to balance maintaining differentiability and maintaining the shape of the bounding box [45]–[47]. Similarly, we use discretely determined masks but ensure that each part of the image maintains gradient flow with Equation 4. By using a small positive value ϵ, this equation results in the lower bound of mask M_{soft} being equal to the small epsilon value whilst maintaining the upper bound of mask M_{discrete}. This soft mask approach ensures gradient flow while maintaining the boolean masking concept.

$$M_{\text{soft}} = M_{\text{discrete}} \cdot (1 - \epsilon) + \epsilon \tag{4}$$

C. Gradient-Descent-Based Optimisation

For our research, we perform constrained gradient-descent-based optimisation. We provide four losses for this optimisation, separated into two categories: constraint losses and optimisable losses. We formulate constraint losses to enforce constraints during the optimisation. The formulation of the optimisable losses is to represent 2D detection performance optimisation.

Constrained Optimisation. For our optimisation procedure, we use an AdamW optimiser [48]. We choose AdamW as this is an adaptive optimiser, so the learning rate is adapted over time to ensure convergence. It also includes momentum and weight decay, ensuring more robustness for handling local optima and the mitigation of overfitting, respectively. To be able to perform batched gradient descent optimisation whilst accounting for computational expenses, we perform all relevant computations for one traffic scene at a time. While this iterative approach may be more time-consuming, it still results in an accumulated gradient based on all traffic scenes in our batch.

To enable constrained optimisation, we use the Basic Differential Multiplier Method (BDMM) [49], enforcing constraints by optimising Lagrange multipliers. The idea is to perform gradient ascent on the Lagrange multipliers, such that multiplying the Lagrange multiplier with the constraint loss results in even higher loss values. This Lagrange multiplication then makes the constraint losses large enough so that the gradient descent optimisation needs to minimise the constraint losses, which is only effectively done by complying with the constraints. We refer to Equation 1 for the constrained loss formation.

Constraint Loss - Out-Of-Bounds loss. Given the prospect of continuous optimisation, we need to account for camera positions remaining in the valid positioning area. The Out-Of-Bounds (OOB) loss penalises camera positions beyond the edges of our vehicle. We define the limits of the vehicle of the ego as $[x_{\text{min}}, x_{\text{max}}]$ and $[z_{\text{min}}, z_{\text{max}}]$. For each camera i, we check its position (x_i, z_i), and if it extends either of the minimum or maximum boundaries, as listed in Equation 5. The ReLU operation ensures we only consider values extending beyond the boundaries. The exponential operation ensures a strict penalisation even for small boundary extensions, as extending beyond the boundary should be discouraged. As the minimum value of the exponential operation is one, we subtract one to ensure that our loss can become zero. We combine these individual camera losses to the final OOB loss L_{oob}, as described in Equation 6. To illustrate the need for such a loss, we refer to Figure 10, representing the final result of a training procedure without any constraint losses. This result has all cameras, represented by yellow dots, extending beyond the bounds of the ego vehicle.

$$\begin{aligned}
L_{oob_{x_{\text{min}}}}(x_i) &= \exp(\text{ReLU}(x_{\text{min}} - x_i)) - 1 \\
L_{oob_{x_{\text{max}}}}(x_i) &= \exp(\text{ReLU}(x_i - x_{\text{max}})) - 1 \\
L_{oob_{z_{\text{min}}}}(z_i) &= \exp(\text{ReLU}(z_{\text{min}} - z_i)) - 1 \\
L_{oob_{z_{\text{max}}}}(z_i) &= \exp(\text{ReLU}(z_i - z_{\text{max}})) - 1
\end{aligned} \tag{5}$$

$$\begin{aligned}
L_{oob_x} &= \sum_i L_{oob_{x_{\text{min}}}}(x_i) + L_{oob_{x_{\text{max}}}}(x_i) \\
L_{oob_z} &= \sum_i L_{oob_{z_{\text{min}}}}(z_i) + L_{oob_{z_{\text{max}}}}(z_i) \\
L_{oob} &= L_{oob_x} + L_{oob_z}
\end{aligned} \tag{6}$$

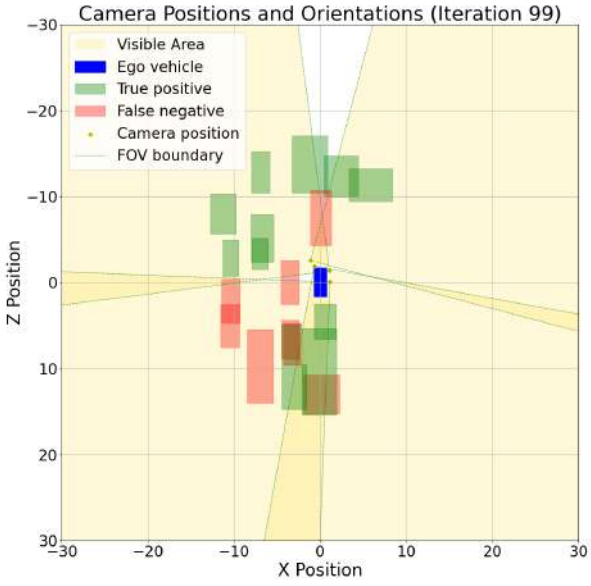


Fig. 10. This figure illustrates the need to account for constraints explicitly. Without the Out-Of-Bounds (OOB) loss, the optimisation pushed all cameras out of the valid positioning area.

Constraint Loss - Camera Distancing Loss. A practical constraint regarding sensor configurations is that sensors can not occupy the same space, requiring a minimum distance between each other. The Camera Distancing (CD) loss penalises cameras for being closer to each other than a minimum threshold. For two arbitrary cameras i and j, we know their absolute

position in our 3D mesh environment as $T[i]$ and $T[j]$. We can determine the three-dimensional Euclidean distance in meters for these two cameras as in Equation 7. To penalise cameras being too close to each other, we formulate the loss per camera pair as in Equation 8, where d_{min} is the minimum distance needed between every two cameras. Given N cameras and $\binom{N}{2}$ unique camera pairs, we formulate the final loss as in Equation 9, as the mean of all pairwise camera losses.

$$d(i,j) = ||T[i] - T[j]||_2 \tag{7}$$

$$L_{\text{CD}}(i,j) = \begin{cases} 0, & d(i,j) \geq d_{\text{min}} \\ \frac{|d(i,j) - d_{\text{min}}|}{d_{\text{min}}}, & \text{else} \end{cases} \tag{8}$$

$$L_{\text{CD}} = \frac{1}{\binom{N}{2}} \sum L_{\text{CD}}(i,j) \tag{9}$$

Optimisable Loss - In-Frame Rotation Loss. To optimise camera positions based on 2D object detection performance, we should ensure that objects are within the camera frame. We use FOV F in radians to sample the rotational space of 2π. We do this by using Equation 2 and Equation 3 from Section B. By sampling the rotational space, we perceive the environment around us from the camera's current position. This rotational sampling means we perceive most of the presence of each non-ego vehicle from this position. By knowing the orientation θ_{i} of camera i, the rotational addition θ_{a}, the width of the image ImgRes_{x}, and the x-value of an in-screen coordinate, we can compute the rotation of this in-frame point $\theta_{if}(x)$ purely from within the image frame, as shown in Equation 10. Suppose we compute this angle for the leftmost and rightmost x-value of the silhouette for all rotational additions a. In that case, we can determine the absolute rotation for the leftmost and rightmost parts of the object as $\theta_{m_{lm}}$ and $\theta_{m_{rm}}$, respectively. If we compare this with θ_{i}, we can compute the rotational difference as described by Equation 11. Knowing the shortest rotational difference for the leftmost and rightmost parts of the object, we can also determine the absolute minimum rotational difference. As we predominantly want to ensure that all vehicles are within the camera frame, we multiply the per-mesh rotational difference with a small positive ϵ value when the object is visible, as described in Equation 12. We determine the smallest rotational difference between each vehicle's correlated cameras. Afterwards, the minimum rotational difference is normalised by π, the largest possible rotational difference, resulting in the per-mesh IFR loss $L_{IFR}(m)$. The final loss is the sum of the per-mesh losses divided by the number of meshes $|M|$, as the mean of the normalised rotational differences for all assigned mesh objects, as described in Equation 13.

$$\theta_{if}(x) = \theta_{\text{i}} + \text{F}\left(\frac{x}{\text{ImgRes}_{\text{x}}} - 0.5\right) \tag{10}$$

$$\begin{aligned} \Delta_{m_{lm}}(i,m) &= \min(|\theta_{m_{lm}} - \theta_i|, 2\pi - |\theta_{m_{lm}} - \theta_i|) \\ \Delta_{m_{rm}}(i,m) &= \min(|\theta_{m_{rm}} - \theta_i|, 2\pi - |\theta_{m_{rm}} - \theta_i|) \end{aligned} \tag{11}$$

$$\begin{aligned} \Delta_{min}(i,m) &= \min(\Delta_{m_{lm}}, \Delta_{m_{rm}}) \cdot S(i,m) \\ S(i,m) &= \begin{cases} \epsilon, & \text{if mesh } m \text{ is visible in camera } i \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

$$\begin{aligned} L_{IFR}(m) &= \frac{\Delta_{min}(i,m)}{\pi} \\ L_{IFR} &= \frac{1}{|M|} \sum_{m \in M} L_{IFR}(m) \end{aligned} \tag{13}$$

Optimisable Loss - Detected Object Area Loss. The Detected Object Area (DOA) loss is our contribution to more directly optimise object detection performance. We use FOV F in radians to sample the rotational space of 2π. We do this by using Equation 2 and Equation 3 from Section B. By sampling the rotational space, we can perceive the environment around us from a camera's current position. This rotational sampling means we also perceive most of the presence of non-ego vehicles from this position. The idea of the DOA loss is that we optimise object visibility, but we imply visibility from detectability. This idea entails that if we assess that a vehicle is detected, we know what the matched bounding box is, and we use this bounding box as the visible area of the object. We compute the visibility percentage of an object mesh m in camera i's camera frame by getting rendered silhouette images of the non-occluded mesh object m, $S_{um}(i,m)$, and one rendered silhouette image of the portion of the non-occluded mesh object m falling inside of its matched bounding box, $S_{dm}(i,m)$. Given these two silhouette image types, we define the visibility percentage computation as given in Equation 14. This formulation entails that lost visibility $V_{doa_{loss}}(i,m)$ equals the difference between the total sum of non-occluded silhouette images in all rotation additions and the sum of pixels outside the detection bounding box. The visibility percentage $V_{doa}(i,m)$ is the difference between the total sum of unblocked silhouette images in all rotation additions and the computed visibility loss. Equation 15 describes loss per mesh $L_{doa}(m)$, where we compute the visibility difference with the minimum amount of visibility $V_{doa_{min}}$. Afterwards, the losses per mesh are summed and normalised to one by dividing by the number of meshes $|M|$ to get L_{doa}, as described in Equation 16.

$$\begin{aligned} V_{doa_{loss}}(i,m) &= \sum_a \sum S_{um}(i,m,\theta_a) - \sum S_{dm}(i,m,\theta_0) \\ V_{doa}(i,m) &= \frac{\sum_a \sum S_{um}(i,m,\theta_a) - V_{doa_{loss}}(i,m)}{\sum_a \sum S_{um}(i,m,\theta_a)} \end{aligned} \tag{14}$$

$$L_{doa}(m) = \begin{cases} 0, & V_{doa_{min}} - V_{doa}(i,m) < 0 \\ \frac{|V_{doa_{min}} - V_{doa}(i,m)|}{V_{doa_{min}}}, & \text{else} \end{cases} \tag{15}$$

$$L_{doa} = \frac{1}{|M|} \sum_{m \in M} L_{doa}(m) \tag{16}$$

IV. Experiments

This section discusses the experiments we performed regarding our proposed method and some comparative baselines. For these experiments, we focus on the following metrics:

- The mean Average Precision at an Intersection over Union (IoU) of 50% (mAP@50). The mAP represents the 2D object detection performance by measuring the precision of object detection at different recall levels. The mAP@50 specifically only considers detection bounding boxes with an IoU of 50% to be True-Positive (TP) detections. We use the mAP@50 to balance good overall detectability without requiring bounding boxes to have near-perfect overlap. For each average mAP@50 value, we consider the mAP values per camera for instances where there are both valid Ground-Truth (GT) bounding boxes and detection model bounding boxes. We average the valid mAP@50 values per camera and retrieve the minimum and maximum average mAP@50 from this. The average mAP@50 is then the average of all camera mAP@50 values.
- The TP detection rate entails how many objects in the test set are correctly detected. We determine that these objects are detected when, for any arbitrary camera, there is a match between a detection bounding box and a GT detection bounding box. While our segregation approach ensures that each vehicle is only associated with the optimisation of one camera, each camera can detect each object.
- The TP in-frame rate entails how many objects in the test set are within the image frame. We determine that an object is present in an image frame if the sum of a correlated silhouette image is non-zero. Like the TP detection rate, each camera can contribute to this metric, even when the object is not part of the camera-specific segregation. In-frame also does not equal visibility, as a (non-)ego vehicle might obstruct the object.
- The detection confidence entails the associated confidence of each correctly detected non-ego vehicle. If multiple cameras detect a particular vehicle, we associate the highest detection confidence with the specific vehicle.
- The optimisable loss entails the loss value on the test set based on the In-Frame Rotation (IFR) and/or Detected Object Area (DOA) losses, so the loss we desire to minimise. Due to different combinations of losses with different scales, this value is only comparable for tests with identically scaled losses.
- The constraint loss entails the loss value on the test set based on the Out-Of-Bounds (OOB) and Camera Distancing (CD) losses, so the loss we desire to eliminate. We scale each constraint loss based on the highest optimisable loss scaling value to ensure compliance with the constraints.
- The minimum camera pair distance entails the smallest distance between any camera pair within the current configuration. This minimum distance verifies the contribution of the CD loss.

Appendix XI shows the parameters each experiment in this section shares. The correlated sections will address the different parameters per experiment series.

Loss function gradient magnitude. To assess the gradients corresponding to our proposed losses, we performed a test where we consider all losses to be unscaled optimisable losses. We then perform per-loss backwards passes and save the corresponding gradients. We did this for ten optimisation runs of 100 iterations each for a traffic scene batch size of one. Table I then shows the L1 average for all non-zero gradient values for both gradients corresponding to positional and rotational movement on the training set. Particular losses only seem to contribute gradients for positional or rotational movement. Unsurprisingly, the Out-Of-Bounds (OOB) and Camera Distancing (CD) loss do not contribute to rotational gradients. However, it may be unexpected that the loss of in-frame rotation (IFR) does not seem to result in positional movement. For the Detected Object Area (DOA) loss, the rotational gradient magnitude seems slightly higher than the positional magnitude.

TABLE I
AVERAGE GRADIENT MAGNITUDE FOR OPTIMISABLE LOSSES

Loss	*Positional*	*Rotational*
OOB	0.977	0.0
CD	0.112	0.0
IFR	0.0	0.353
DOA	0.003	0.016

Gradient-Based Optimisation. To assess the performance of our proposed losses, we performed four experiments. For each experiment, we performed five optimisation runs for 100 iterations with four 90-degree cameras and a batch size of four. We started with separate experiments for the In-Frame Rotation (IFR) and Detected Object Area (DOA) losses. Afterwards, we combine both losses to see if the combination improves optimisation. Lastly, we scale the DOA loss by 10 based on the gradient magnitudes from Table I, ensuring the average gradient magnitudes are more in line.

The red "SGD" (Stochastic Gradient Descent) part of Table II shows the results of the four experiments. Combining the IFRL and the scaled DOA loss results in better average values for the most important metrics: the mean Average Precision (mAP) and the True-Positive (TP) detection rate. We note that only using the DOA loss generally results in the highest value for the maximum mAP for a camera. However, IFR loss seems to be needed to ensure objects are within the image frame, as the test run with only the DOA loss has the lowest TP detection rate.

The red "SGD" part of Table III shows results related to the constraints. Since the constraint losses are all zero,

TABLE II
TEST SET RESULTS FOR STOCHASTIC GRADIENT DESCENT (SGD), RANDOM SAMPLING STRATEGY (RANDOM), AND HIGHEST POINT ON VEHICLE (HIGHEST)

Experiment		*Average mAP@50*		*Min Camera mAP@50*		*Max Camera mAP@50*		*TP Detection Rate*		*TP In-Frame Rate*		*Detection Confidence*		*Optimisable Loss*	
Method	*Loss*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*
SGD	***IFR***	0.4801	0.0689	0.2860	0.2615	0.5935	0.0597	0.7514	0.1276	0.9988	**0.0011**	0.6079	0.0176	0.0003	0.0002
	DOA	0.5258	0.0494	0.4304	0.1118	**0.6306**	0.0392	0.6274	0.0922	0.7969	0.0991	0.6245	0.0319	0.4258	0.0759
	IFR+DOA	0.5262	0.0301	0.4350	0.0865	0.5946	0.0372	0.7918	0.0703	0.9992	**0.0011**	0.6142	0.0533	0.2578	0.0615
	IFR+10*DOA	**0.5537**	0.0340	**0.5073**	0.0279	0.6099	0.0478	0.8518	**0.0393**	0.9847	0.0155	0.6061	0.0167	2.0094	0.5328
Random	***IFR+DOA***	0.5340	0.0374	0.4657	0.0908	0.5949	0.0550	0.7737	0.0863	0.9035	0.0960	0.6225	0.0249	0.3139	0.1289
	IFR+10*DOA	0.5071	**0.0216**	0.4566	**0.0263**	0.5583	**0.0154**	0.7470	0.1089	0.9113	0.0361	0.6282	**0.0115**	2.3112	0.7461
Highest	***IFR+DOA****	0.5131	*N/A*	0.4665	*N/A*	0.5824	*N/A*	**0.8961**	*N/A*	**1.0**	*N/A*	**0.6414**	*N/A*	0.1241	*N/A*

TABLE III
AVERAGE GRADIENT MAGNITUDE FOR OPTIMISABLE LOSSES

Experiment		*Constraint Loss*		*Min Camera Pair Distance (m)*	
Method	*Loss*	*Mean*	*STD*	*Mean*	*STD*
SGD	***IFR***	0.0	0.0	0.3816	0.2045
	DOA	0.0	0.0	0.5190	0.1746
	IFR+DOA	0.0	0.0	0.5778	0.2713
	IFR+10*DOA	0.0	0.0	0.5321	0.2942
Random	***IFR+DOA***	0.0	0.0	0.5735	0.2211
	IFR+10*DOA	0.0	0.0	0.3612	0.0634
Highest	***IFR+DOA***	0.0	*N/A*	0.2000	*N/A*

the constrained optimisation procedure with the Lagrange multipliers resulted in satisfied constraints.

Figure 11 visually represents the result of an optimisation procedure. Starting from an arbitrary set of camera parameters, as on the left, the gradient-descent-based optimisation results in the camera configuration in the middle of the figure. The right part of the figure shows how this optimisation correlates to test set performance.

Random Sampling Strategy. As a comparison strategy to our proposed gradient-descent-based method, we use a random sampling strategy. We sample a random set of camera positions for each iteration and compute the loss for the batch of traffic scenarios. We then select the camera configuration corresponding to the iteration with the lowest training loss and use this configuration to evaluate based on the test set. To ensure we comply with our desired constraints, we explicitly pick the lowest loss with a constraint loss of zero. We perform two experiments: one for the unscaled combination of the IFR and DOA losses and one for the combination of IFR loss with the scaled DOA loss. We perform five optimisation runs for each experiment with the same settings as the gradient-descent-based method.

The green "Random" part of Table II shows the average results of the two experiments. The results for randomly sampled positions are pretty good for both loss combinations. While there is likely some discrepancy between the performance on a subset of the training set and the performance on the entire test set performance, we initially expected this to be a more significant issue than it ended up being.

The green "Random" part of Table III shows results related to the constraints. Since we explicitly only consider configurations that satisfy the constraints, the average constraint loss results in zero. Interestingly, the experiment with the scaled DOA loss resulted in a substantially lower standard deviation for the minimum camera pair distance.

Highest point testing. Combined with the random sampling approach, we also provide a more intuitive baseline. When placing a sensor configuration consisting of four cameras on a vehicle, the highest point on the vehicle's roof is a prominently used position. Research optimising the position of LiDAR sensors also indicated that the preferred configuration seems to be around this highest point [6]. Therefore, we propose to have a baseline configuration around the highest point on the ego vehicle. We determine the highest point on our ego vehicle and position the cameras to ensure the desired minimum camera distance. Regarding orientation, we orient each camera with 90-degree intervals, ensuring that one camera is at least facing forward.

The blue "Highest" part of Table II shows the average results for one test iteration. Note the "N/A" values for the standard deviation, as it is a singular deterministic configuration with only one performed test run. This baseline results in a good TP detection rate. While this method does not use a loss, we provide the optimisation loss value to show the hypothetical optimisation value at this position. This loss shows that the combination of the In-Frame Rotation (IFR) and Detected Object Area (DOA) losses result in a low loss for this sensor configuration.

The blue "Highest" part of Table III shows results related to the constraints. We exactly comply with the constraints since we deterministically position the cameras to account for the desired minimum camera distance.

Figure 12 shows the results on the test set of this intuitive baseline. The yellow visible area shows that the rotational space is fully covered.

V. Discussion

Our experiments in Section IV show that for the tested gradient-descent-based methods, the combination of the In-Frame Rotation (IFR) loss and a scaled Detected Object Area (DOA) loss seems to result in the best mean minimum Average Precision (mAP) and True Positive (TP) detection rate. Considering that only using the IFR loss essentially entails a random position while only optimising the rotation,

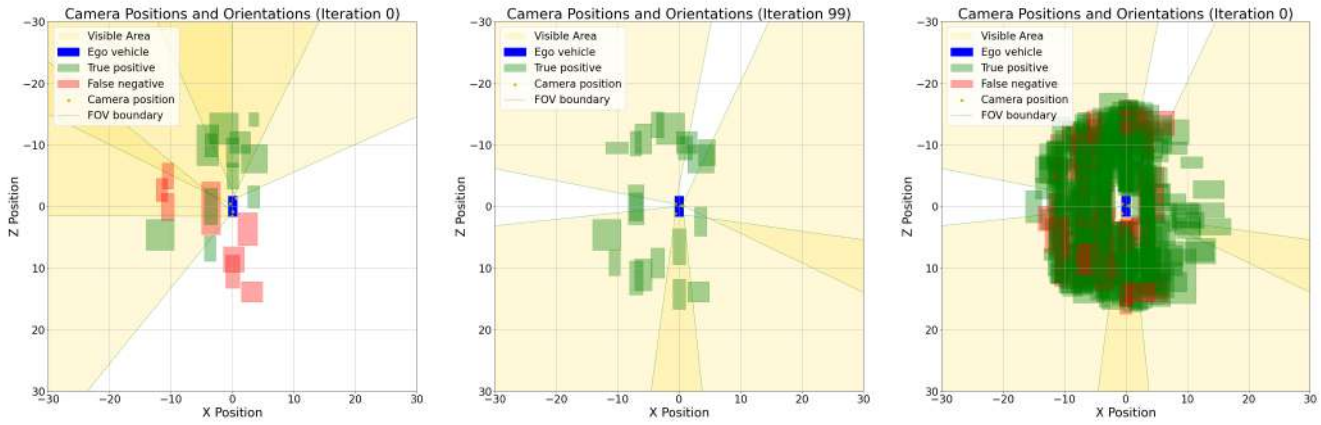


Fig. 11. The figure shows three images of the gradient-descent-based optimisation procedure. The left image shows the first iteration of the training procedure, where the camera positions are random. The middle image shows the final iteration of the training procedure, showing that all vehicles belonging to the current batch are detected. The right-most image shows the performance of the final position on the entire test set. The red rectangles in all images represent False Positive vehicle detections, whereas the green rectangles represent True Positive detections.

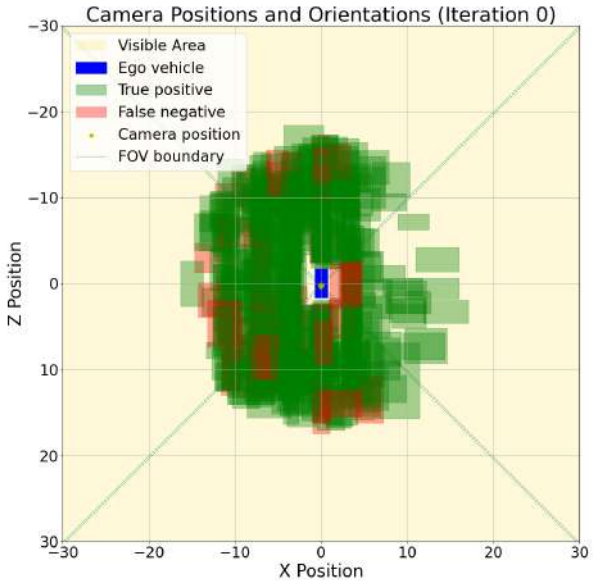


Fig. 12. The figure shows the performance of the highest position on the ego vehicle on the entire test set. The red rectangles in all images represent False Positive vehicle detections, whereas the green rectangles represent True Positive detections.

this loss provides better results than initially expected. Interestingly, the increase of the average mAP@50 seems to result from maintaining a consistently higher per-camera mAP@50 lower bound rather than substantially increasing the maximum per-camera mAP@50. The optimisation limiting the upper mAP@50 per-camera bound is likely partially caused by our environment representation, camera parameters, or the lower detection confidence bound.

When comparing our best gradient-descent-based optimisation result with the random sampling strategy, our method provides better mean values regarding the mAP@50, the TP detection rate and the TP in-frame rate. The random sampling approach results in higher detection confidence, which could stem from the number of TP detections and lower bound detection confidence. When our method tries to maximise the amount of TP detections, it likely also accumulates more lower-confidence detections. Considering a higher confidence threshold for our approach might lead to more high-confidence detections. In general, the random sampling strategy still holds up surprisingly well for both loss combinations. The context of our optimisation problem, with four cameras with a 90-degree FOV and a favourable height offset, might be relatively easy to satisfy, even for 100 randomly sampled configurations. We also expected more of a disconnect between the lowest loss on a particular training batch and the loss on the testing set.

When comparing our best gradient-descent-based optimisation result with the highest point on the vehicle, our method still provides better mean values regarding the mAP@50. However, the intuitive baseline is better regarding the TP detection rate, the TP in-frame rate and the detection confidence. The confidence value is interesting, as this might indicate a slight bias to this highest position, being more in line with traditional camera configurations on an AV. The TP detection and in-frame rates could also stem from the higher position that maximises environmental coverage. For our research, we should consider why our optimisation does not push the cameras to a similar position. Due to gradient-based optimisation, a likely culprit is that optimisation pushes us to a local optimum instead of being pushed to the global optimum. This discrepancy can also stem from some of our optimisation-related parameters, like the minimum object visibility threshold, the subsampling approach, or the batch size. For the visibility threshold, ensuring this visibility for all vehicles might push the cameras to a position with less general environmental coverage. Our per-camera object segregation

approach might also result in less generalisability because each camera focuses on a specific cluster of vehicles in a particular direction.

While our best gradient-descent-based optimisation result does not ensure optimisation on all metrics we discuss, the improved mAP performance is the more important. While the TP detection rate is important to consider, the metric only really considers that there is a detection bounding box that matches a Ground-Truth (GT) bounding box. The mAP value considers TP detections, False Positive (FP) detections and False Negative (FN) detections. These additional considerations make mAP a more insightful metric for downstream task performance.

VI. Future work

This section discusses potential future continuations of our research. A critical need for improvement is the valid positioning area, as limiting this area to the roof and hood of the vehicle limits the optimisation. In principle, the sensor can be anywhere on the vehicle's surface, and sensor placement optimisation should account for that. However, then the question becomes how we can represent this surface within the context of continuous optimisation. Sensor placement optimisation strategies should consider more than only two positional and one rotational Degree of Freedom (DoF). The choice of optimisable parameters directly aligns with the sensor positioning method.

While our traffic scene reconstructions with separable mesh vehicles contributed significantly to the results of this project, it is hard to argue about their visual fidelity. Given our proposed method, it is possible to provide better-looking environments, as illustrated in Figure 13. However, one must account for additional computational expenses. The rise of implicit representations over the last few years, like with NeRF [22] and 3D Gaussian Splatting [23], provide interesting options to represent photo-real environments while generally limiting the computational expenses needed to render them. However, the optimisation of the implicit environment generally requires higher computational expenses, mitigated by efficient environment sampling or hardware acceleration. Additionally, one must account for object separation from their respective environments for integration with this project.

Regarding the optimisation procedure, our approach could be even more related to actual downstream task performance. While we strive to incorporate downstream task performance in our optimisation, our "visibility through detectability" approach is still a surrogate metric. The ideal scenario would be integrating an arbitrary detection model with gradient flow in the optimisation loop and directly optimising the bounding box loss, the classification loss, or the objectness loss rather than using handcrafted metrics. However, the main issue with such an approach is that this optimisation remains limited to what a camera sees, and in scenarios without objects, this could result in no optimisation at all. With our idea of sampling the 360-degree rotational space with the camera's FOV, we can assess where objects in the environment are and directly point the camera to them. To account for occlusions, we could assess the detectability of singular vehicles and all vehicles simultaneously. Then, we can infer the detection performance of the collective traffic scene by correlating it to the detectability of the singular vehicle.

Fig. 13. The figure illustrates a triangle mesh representation of a traffic scene without simplifying the point clouds and triangle meshes. Not simplifying these representations makes the environment around the vehicles substantially less noisy compared to our mesh representations but increases computational expenses with the differentiable renderer.

Lastly, another research direction could be integrating different or multiple sensor types into the optimisation, like LiDAR and radar. Incorporating these sensors would require a different approach than a camera, as these two sensors result in a point cloud rather than a 2D image. If a differentiable renderer allows for a point cloud rendering, this could allow for a more direct integration of the sensors. Another implicit approach would be to represent the performance of a particular downstream task via the object in the differentiable renderer, similar to our Detected Object Area (DOA) approach. This more implicit representation might be more suitable for heterogeneous sensor configurations, where the detectability of an object can rely on the detection performance of each sensor or the combined sensor configuration. Regarding object detection, we should consider which vehicles must be detected. Rather than focussing on detecting all vehicles, we could prioritise detecting certain vehicles. For instance, vehicles close to the ego vehicle should have a higher prioritisation compared to vehicles further away. Additionally, research has proposed a metric that determines an intuitive ranking of how important the detecting of each vehicle in an environment is [50], potentially providing an interesting way to retrieve task-dependant importance of to-be detected objects.

VII. Conclusion

This research proposes a sensor placement optimisation strategy to address the research gap of more explicit downstream task optimisation. In our approach, we combine gradient-descent-based optimisation with a differentiable renderer to

optimise the position of cameras on an Autonomous Vehicle (AV) on the downstream task of object detection. To enable this optimisation, we propose creating mesh representations of traffic scenes from an AV simulator to represent in the differentiable renderer. With our approach, we present four loss functions. Two constraint losses ensure camera positions within the valid positioning area and minimum camera distance between each two cameras. Two optimisation losses minimise the rotational difference between the camera orientation and the rotation needed to be in the frame and minimise the difference between a minimum detected object area and a set minimum threshold. We compare the best loss combination for our approach with a random sampling strategy and an intuitive best position around the highest point on the ego vehicle. For both comparisons, our approach results in improved mAP@50 values. This comparison expresses the value of explicitly accounting for downstream task performance during sensor placement optimisation.

References

[1] H. A. Ignatious, Hesham-El-Sayed, and M. Khan, "An overview of sensors in autonomous vehicles," *Procedia Computer Science*, vol. 198, pp. 736–741, 2022, 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921025540

[2] H. Hu, Z. Liu, S. Chitlangia, A. Agnihotri, and D. Zhao, "Investigating the impact of multi-lidar placement on object detection for autonomous driving," 2022. [Online]. Available: https://arxiv.org/abs/2105.00373

[3] Y. Li, L. Kong, H. Hu, X. Xu, and X. Huang, "Is your lidar placement optimized for 3d scene understanding?" 2024. [Online]. Available: https://arxiv.org/abs/2403.17009

[4] V. A. Puligandla and S. Lončarić, "A continuous camera placement optimization model for surround view," *IEEE Transactions on Intelligent Vehicles*, pp. 1–11, 2023.

[5] J. Dey, W. Taylor, and S. Pasricha, "Vespa: A framework for optimizing heterogeneous sensor placement and orientation for autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 10, no. 2, pp. 16–26, 2021.

[6] F. Berens, S. Elser, and M. Reischl, "Genetic algorithm for the optimal lidar sensor configuration on a vehicle," *IEEE Sensors Journal*, vol. 22, no. 3, pp. 2735–2743, 2022.

[7] C. M. Costa, G. Veiga, A. Sousa, U. Thomas, and L. Rocha, "Sensor placement optimization using random sample consensus for best views estimation," in *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2023, pp. 29–36.

[8] J. Spruit and D. Gavrilla, "Ultrasonic perception for autonomous vehicles," 08 2023. [Online]. Available: http://resolver.tudelft.nl/uuid:bce22af4-61b1-4cf6-85a7-0ff960a690ab

[9] E. Arnold, S. Mozaffari, M. Dianati, and P. A. Jennings, "Visual sensor pose optimisation using rendering-based visibility models for robust cooperative perception," *CoRR*, vol. abs/2106.05308, 2021. [Online]. Available: https://arxiv.org/abs/2106.05308

[10] J. Dybedal and G. Hovland, "Gpu-based optimisation of 3d sensor placement considering redundancy, range and field of view," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020, pp. 1484–1489.

[11] V. Akbarzadeh, J.-C. Levesque, C. Gagné, and M. Parizeau, "Efficient sensor placement optimization using gradient descent and probabilistic coverage," *Sensors (Basel, Switzerland)*, vol. 14, pp. 15 525 – 15 552, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:313315

[12] N. Ravi, J. Reizenstein, D. Novotný, T. Gordon, W. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with pytorch3d," *CoRR*, vol. abs/2007.08501, 2020. [Online]. Available: https://arxiv.org/abs/2007.08501

[13] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 11 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[15] Scratchapixel. (2023) Rasterization. [Online]. Available: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/overview-rasterization-algorithm.html

[16] S. Liu, W. Chen, T. Li, and H. Li, "Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction," *CoRR*, vol. abs/1901.05567, 2019. [Online]. Available: http://arxiv.org/abs/1901.05567

[17] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," 2017.

[18] W. Chen, J. Gao, H. Ling, E. J. Smith, J. Lehtinen, A. Jacobson, and S. Fidler, "Learning to predict 3d objects with an interpolation-based differentiable renderer," *CoRR*, vol. abs/1908.01210, 2019. [Online]. Available: http://arxiv.org/abs/1908.01210

[19] J. U. Müller, M. Weinmann, and R. Klein, "Unbiased gradient estimation for differentiable surface splatting via poisson sampling," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 281–299.

[20] G. Loubet, N. Holzschuch, and W. Jakob, "Reparameterizing discontinuous integrands for differentiable rendering," *ACM Trans. Graph.*, vol. 38, no. 6, 11 2019. [Online]. Available: https://doi.org/10.1145/3355089.3356510

[21] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," *ACM Trans. Graph.*, vol. 37, no. 6, 12 2018. [Online]. Available: https://doi.org/10.1145/3272127.3275109

[22] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020.

[23] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 7 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[24] Y. Sun, X. Wang, Y. Zhang, J. Zhang, C. Jiang, Y. Guo, and F. Wang, "icomma: Inverting 3d gaussians splatting for camera pose estimation via comparing and matching," 2023.

[25] Ágoston István Csehi and C. M. Józsa, "Bid-nerf: Rgb-d image pose estimation with inverted neural radiance fields," 2023.

[26] Y. Lin, T. Müller, J. Tremblay, B. Wen, S. Tyree, A. Evans, P. A. Vela, and S. Birchfield, "Parallel inversion of neural radiance fields for robust pose estimation," 2023.

[27] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone, "Loc-nerf: Monte carlo localization using neural radiance fields," 2022.

[28] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "Inerf: Inverting neural radiance fields for pose estimation," 2021.

[29] J. Tremblay, B. Wen, V. Blukis, B. Sundaralingam, S. Tyree, and S. Birchfield, "Diff-dope: Differentiable deep object pose estimation," 2023.

[30] S. H. Bengtson, H. Åström, T. B. Moeslund, E. A. Topp, and V. Krueger, "Pose estimation from rgb images of highly symmetric objects using a novel multi-pose loss and differential rendering," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4618–4624.

[31] D. Beker, H. Kato, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Monocular differentiable rendering for self-supervised 3d object detection," *CoRR*, vol. abs/2009.14524, 2020. [Online]. Available: https://arxiv.org/abs/2009.14524

[32] Y. Shibaike and K. Iwasaki, "Optical parameter estimation for hair and fur using differentiable rendering," in *SIGGRAPH Asia 2022 Technical Communications*, ser. SA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3550340.3564221

[33] G. Juglan, "Solving the art gallery problem using gradient descent," 2022. [Online]. Available: https://studenttheses.uu.nl/handle/20.500.12932/43207

[34] N. Petruzelli, "How to guard an art gallery: A simple mathematical problem," *The Review: A Journal of Undergraduate Student Research*, vol. 23, 2022. [Online]. Available: https://fisherpub.sjf.edu/ur/vol23/iss1/7

[35] Y. Ma, Y. B. Zheng, S. Y. Wang, Y. D. Wong, and S. M. Easa, "Virtual-real-fusion simulation framework for evaluating and optimizing small-spatial-scale placement of cooperative roadside sensing units," *Computer-Aided Civil and Infrastructure Engineering*, vol. n/a, no. n/a, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.13167

[36] T. Ma, Z. Liu, and Y. Li, "Perception entropy: A metric for multiple sensors configuration evaluation and design," 2021. [Online]. Available: https://arxiv.org/abs/2104.06615

[37] S. Jin, Y. Gao, F. Hui, X. Zhao, C. Wei, T. Ma, and W. Gan, "A novel information theory-based metric for evaluating roadside lidar placement," *IEEE Sensors Journal*, vol. 22, no. 21, pp. 21 009–21 023, 2022.

[38] CARLA. (2025) Sensors reference - Depth camera. [Online]. Available: https://carla.readthedocs.io/en/latest/ref_sensors/#depth-camera

[39] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural kernel surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4369–4379.

[40] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06. Goslar, DEU: Eurographics Association, 2006, p. 61–70.

[41] F. Williams, Z. Gojcic, S. Khamis, D. Zorin, J. Bruna, S. Fidler, and O. Litany, "Neural fields as learnable kernels for 3d reconstruction," *CoRR*, vol. abs/2111.13674, 2021. [Online]. Available: https://arxiv.org/abs/2111.13674

[42] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, S. Zhao, S. Cheng, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2020. [Online]. Available: https://arxiv.org/abs/1912.04838

[43] A. Muntoni and P. Cignoni, "PyMeshLab," Jan. 2021.

[44] M. Garland and P. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics," in *Proceedings Visualization '98 (Cat. No.98CB36276)*, 1998, pp. 263–269.

[45] T. Konishi, M. Kurokawa, C. Ono, Z. Ke, G. Kim, and B. Liu, "Parameter-level soft-masking for continual learning," 2023. [Online]. Available: https://arxiv.org/abs/2306.14775

[46] A. Athar, J. Luiten, A. Hermans, D. Ramanan, and B. Leibe, "Differentiable soft-masked attention," 2022. [Online]. Available: https://arxiv.org/abs/2206.00182

[47] A. Rochow, M. Schwarz, M. Weinmann, and S. Behnke, "Fadiv-syn: Fast depth-independent view synthesis using soft masks and implicit blending," 2022. [Online]. Available: https://arxiv.org/abs/2106.13139

[48] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[49] J. C. Platt and A. H. Barr, "Constrained differential optimization," in *Proceedings of the 1st International Conference on Neural Information Processing Systems*, ser. NIPS'87. Cambridge, MA, USA: MIT Press, 1987, p. 612–621.

[50] J. Philion, A. Kar, and S. Fidler, "Learning to evaluate perception models using planner-centric metrics," 2020. [Online]. Available: https://arxiv.org/abs/2004.08745

VIII. Appendix: Environment Creation Images

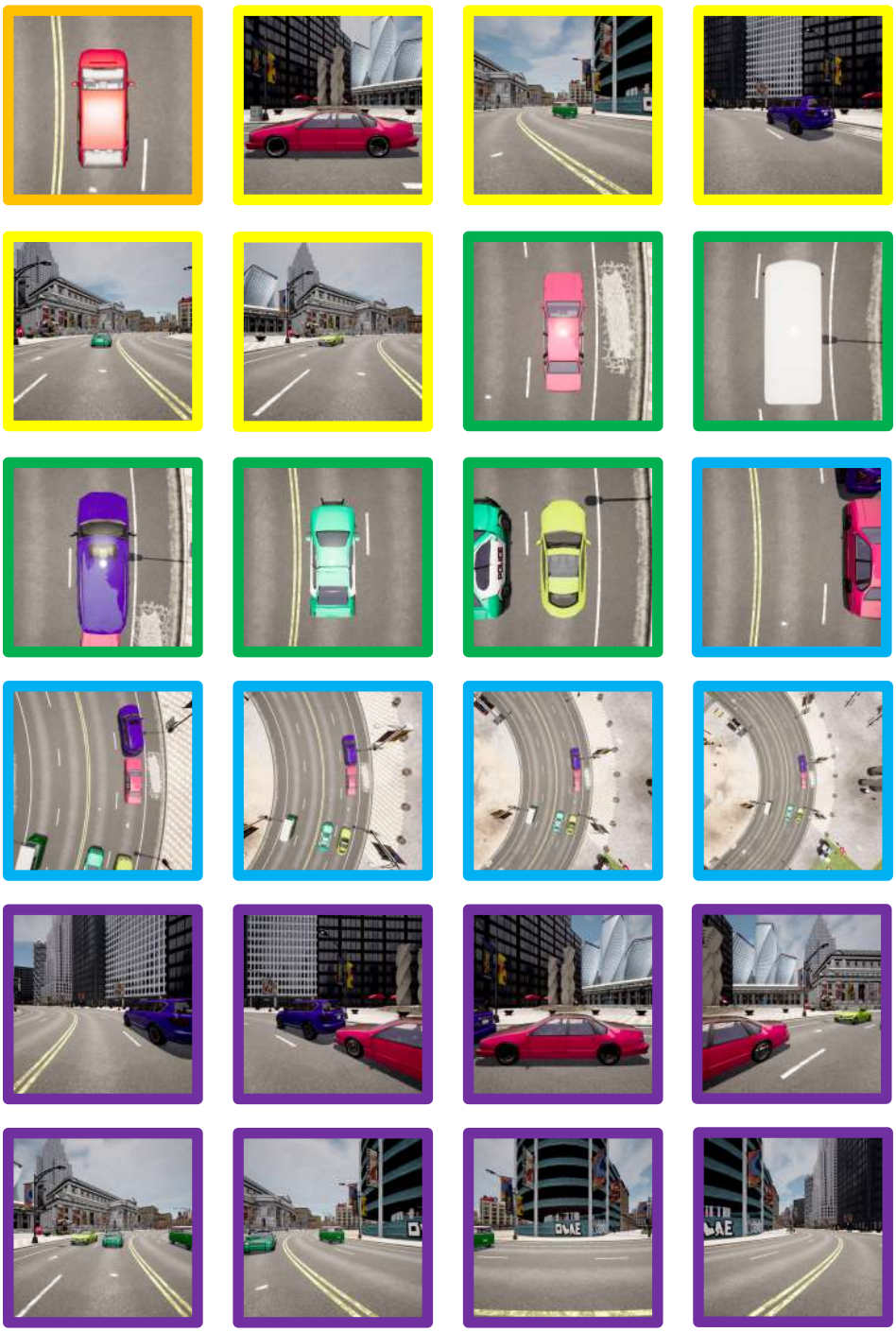


Fig. 14. The following 24 images are the RGB images from a singular traffic scene. The orange image is a top-down image of the ego vehicle. The yellow images are images from the ego vehicle's position directly pointed toward the other vehicles in the environment. The green images are top-down images of these non-ego vehicles. The blue images are multiple top-down images from the ego vehicle location. The purple images are multiple rotations from the position of the ego vehicle.

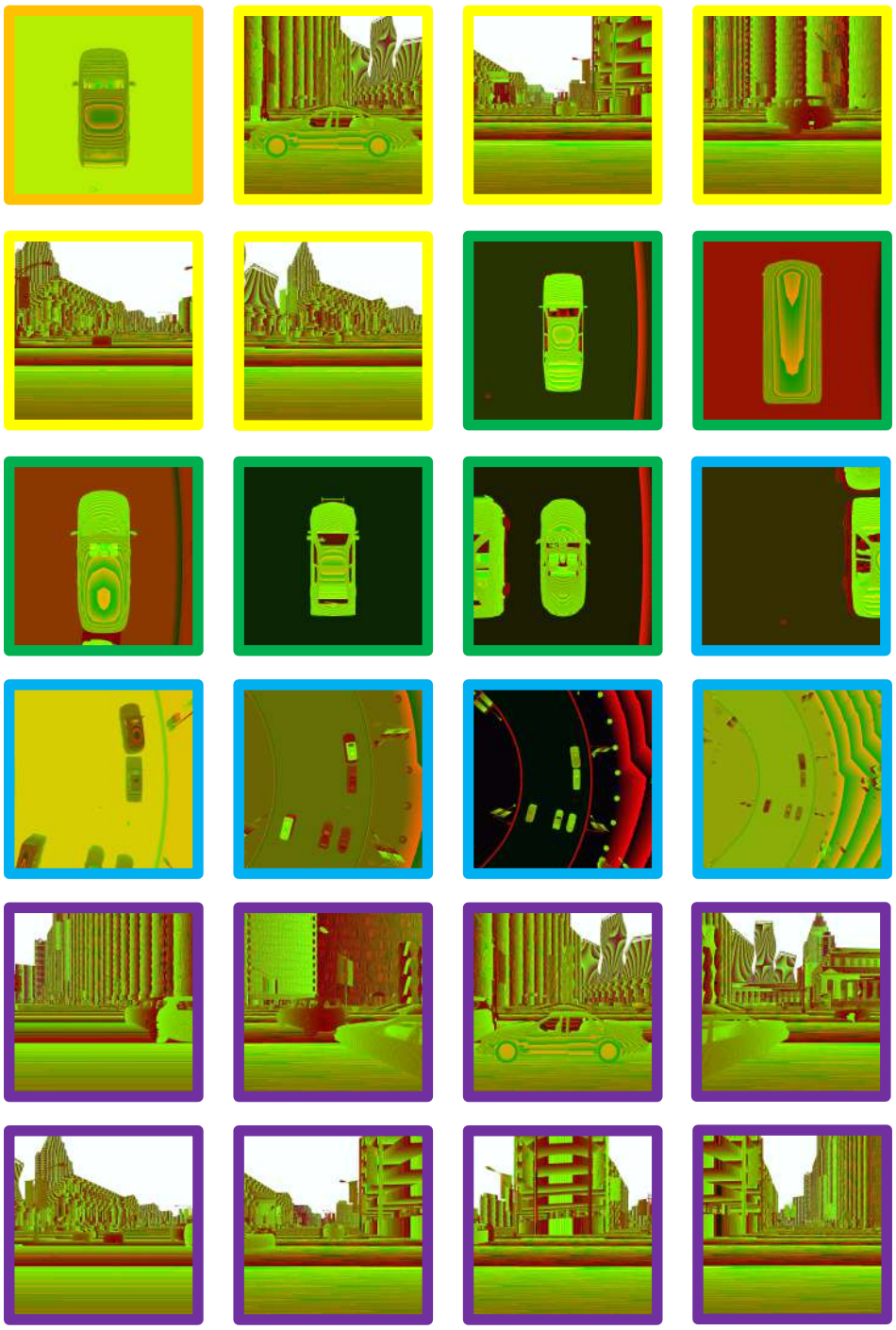


Fig. 15. The following 24 images are the depth images from a singular traffic scene. The orange image is a top-down image of the ego vehicle. The yellow images are images from the ego vehicle's position directly pointed toward the other vehicles in the environment. The green images are top-down images of these non-ego vehicles. The blue images are multiple top-down images from the ego vehicle location. The purple images are multiple rotations from the position of the ego vehicle.

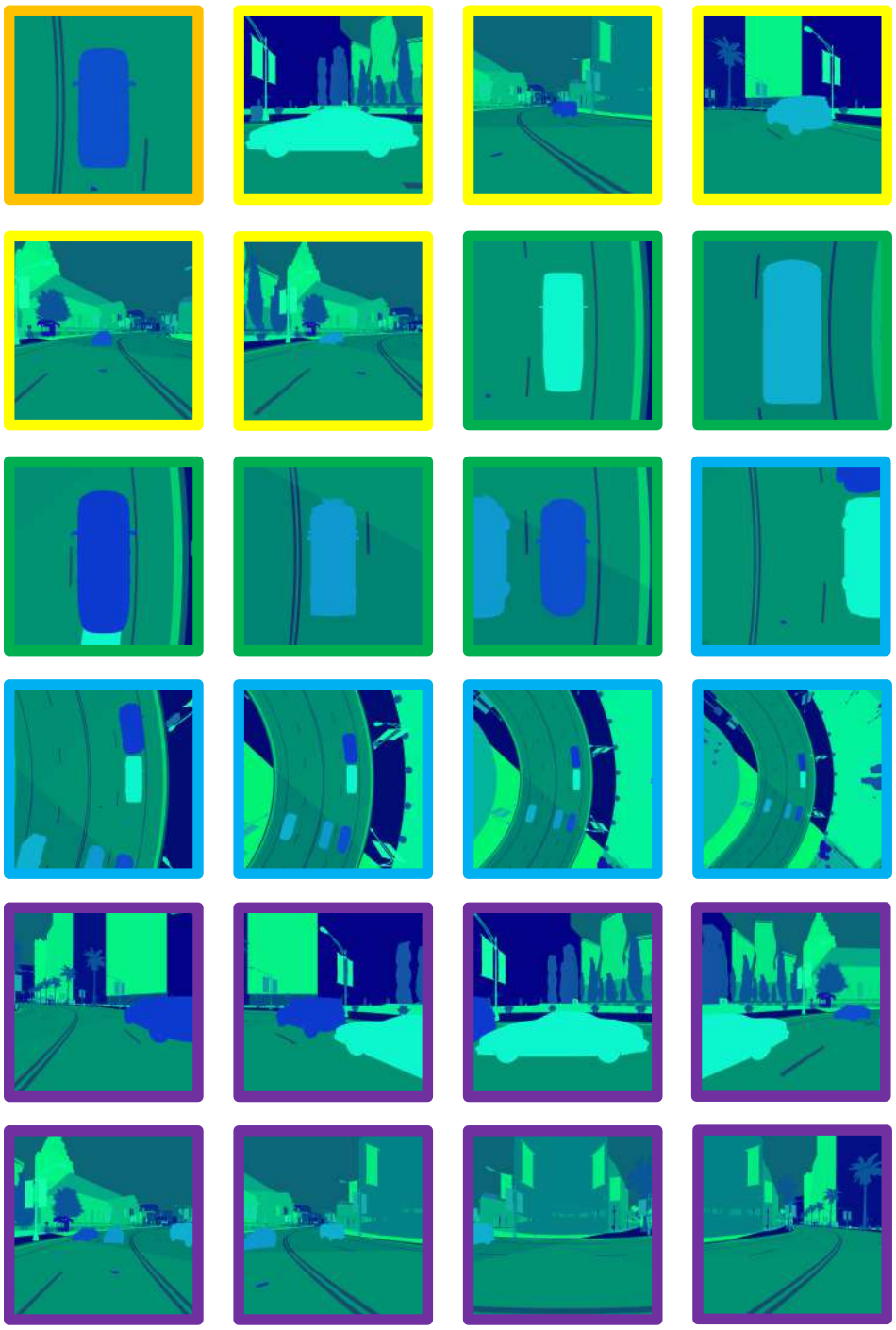


Fig. 16. The following 24 images are the instance images from a singular traffic scene. The orange image is a top-down image of the ego vehicle. The yellow images are images from the ego vehicle's position directly pointed toward the other vehicles in the environment. The green images are top-down images of these non-ego vehicles. The blue images are multiple top-down images from the ego vehicle location. The purple images are multiple rotations from the position of the ego vehicle.

IX. Appendix: Cut Losses

This section covers some of the losses that were not used for the main body of this document for various reasons.

A. Surface Blockage Loss

From experience, we noticed that the ego vehicle could block non-ego vehicles in the environment, hamper optimisation. Therefore, we specifically introduce a Surface Blockage (SB) loss to help during such occurrences. We compute the visibility percentage of an object mesh m in camera i's camera frame by getting a rendered silhouette image of the non-occluded mesh object m, $S_{um}(i,m)$, a rendered silhouette image of the ego vehicle, $S_e(i,m)$, and a rendered silhouette image of the two objects combined in the same scene, $S_{m+e}(i,m)$. Given these three silhouette images, we compute the visibility percentage as in equation 17. For the largest visibility percentage, we compare it with fixed hyperparameter $V_{SB_{min}}$. $V_{SB_{min}}$ denotes the minimum visibility the SB loss should ensure. We compute the SB loss as in equation 18. Afterwards, the losses per mesh are summed and normalised to one by dividing by the number of meshes $|M|$ as described in 19. We did not end up using this loss as, while we could occasionally link a poor optimisation performance to a higher SB loss value, optimising this loss generally worsened overall optimisation.

$$V_{SB}(i,m) = \frac{\sum S_{m+e}(i,m) - \sum S_e(i)}{\sum S_m(i,m)} \tag{17}$$

$$L_{SB}(m) = \begin{cases} 0, & V_{SB_{min}} - V_{SB}(m) < 0 \\ \frac{|V_{SB_{min}} - V_{SB}(m)|}{V_{SB_{min}}}, & \text{else} \end{cases} \tag{18}$$

$$L_{SB} = \frac{1}{|M|} \sum_{m \in M} L_{SB}(m) \tag{19}$$

B. Angular Coverage Loss

To ensure that the entire rotational space was covered as much as possible, we propose an Angular Coverage (AC) loss. The maximum rotational visibility for N cameras equals N cameras times the FOV F radians until the maximum possible coverage of 2π radians, as illustrated in Equation 20. Each camera i covers a range based on the current plane rotation θ_i and the FOV F, as illustrated by Equation 21. Computing the unique rotational coverage requires accounting for the overlap of rotational ranges. To account for this, we first sort all ranges by the starting angle from small to large. Afterwards, we normalise the smallest starting angle of a range to zero, and all other angles are normalised relative to this smallest angle in the $[0, 2\pi]$ range. If a range starts before 2π and ends after 0, we split it into two separate ranges. Next, we iterate over each range. If an angular range overlaps with the previous range, we only add the unique coverage from this range. If this range does not overlap with the previous range, we add the entire coverage from this range. Mathematically, this describes the union of all ranges, so the unique coverage of the rotational space, as described in Equation 22. For the final loss, we take the difference between the maximum possible coverage C_{max} and the unique coverage C_{union} from all cameras. Afterwards, we divide this by the difference of maximum coverage C_{max} and C_{fov}, which is the inherent coverage of one camera which equals the FOV F in radians. As the worst coverage cannot be less than the coverage of one camera, we divide it by this difference to have a more valuable interpretation of a loss of one, as described by Equation 23. We did not end up using the AC loss, as while ensuring full rotational space coverage may be important, it may be too limiting for our tests with a maximum of four cameras.

$$C_{\text{max}} = \min(N \cdot F, 2\pi) \tag{20}$$

$$C_i = \left[\theta_i - \frac{F}{2}, \theta_i + \frac{F}{2}\right] \tag{21}$$

$$C_{\text{union}} = \bigcup_{i=0}^{N} C_i \tag{22}$$

$$L_{\text{ac}} = \frac{C_{\text{max}} - C_{\text{union}}}{C_{\text{max}} - C_{\text{fov}}} \tag{23}$$

C. Object Visibility Loss

The Object Visibility (OV) loss is quite similar to the Detected Object Area (DOA) loss, but instead of optimising visibility through detectability, it optimises actual visibility. Equation 24 shows this formulation as the percentage of visible vehicle presence in the primary camera orientation based on the sum of non-blocked vehicle presence in all rotational additions θ_a. For the main structure of this loss, we refer back to the DOA loss formulation in Section C. The main idea for this loss was to provide comparison material for a more downstream task-oriented loss, but unfortunately, we lacked the time to perform tests with this.

$$\begin{aligned} V_{ov_{loss}}(i,m) &= \sum_a \sum S_{um}(i,m,\theta_a) - \sum S_{bm}(i,m,\theta_0) \\ V_{ov}(i,m) &= \frac{\sum_a \sum S_{um}(i,m,\theta_a) - V_{ov_{loss}}(i,m)}{\sum_a \sum S_{um}(i,m,\theta_a)} \end{aligned} \tag{24}$$

X. Appendix: Cut Losses Experiments

This section covers some limited experiments that were performed with some of the cut losses. Table V presents the gradient magnitudes of all three cut losses: the Angular Coverage (AC) loss, the Surface Blockage (SB) loss and the Object Visibility (OV) loss. Table IV present some results where the SB and OV losses are included. Any results for constraint losses comply with the conclusions of the main body, and therefore are excluded here.

TABLE IV
APPENDIX TEST SET RESULTS FOR STOCHASTIC GRADIENT DESCENT (SGD)

Experiment		*Average mAP@50*		*Min Camera mAP@50*		*Max Camera mAP@50*		*TP Detection Rate*		*TP In-Frame Rate*		*Detection Confidence*		*Optimisable Loss*	
Method	*Loss*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*	*Mean*	*STD*
SGD	***IFR+DOA+SB***	0.5091	0.0548	0.3513	0.1815	0.6288	0.0370	0.7404	0.1012	0.9988	0.0017	0.5790	0.0263	0.3892	0.1828
	IFR+DOA+SB+AC	0.4980	0.0357	0.4121	0.0834	0.5626	0.0268	0.7588	0.1628	1	0	0.5795	0.0670	0.3864	0.2988

TABLE V
AVERAGE GRADIENT MAGNITUDE FOR OPTIMISABLE LOSSES

Loss	*Positional*	*Rotational*
AC	0.0	2.116
SB	0.002	0.010
OV	0.005	0.026

XI. Appendix: Experiment Parameters

For the experiments in Section IV, we make use of the following global settings:

- Optimisation loss optimiser starting learning rate of 0.25.
- Optimisation loss optimiser starting weight decay of 1e-3.
- Constraint loss optimiser starting learning rate of 0.25.
- Constraint loss optimiser starting weight decay of 1e-3.
- 100 iterations per training run.
- Image resolution of 256x256.
- A small epsilon value of 1e-5.
- The per-camera object segregation approach does 16 rotations at the (0.0, 0.0, 0.0) coordinate.
- The minimum amount of Detected Object Area (DOA) visibility $V_{doa_{min}}$ of 50%.
- Minimum camera pair distance of 0.2 metres.
- Height offset of 0.25 metres.
- Four cameras with a Field Of View (FOV) of 90 degrees.
- YOLO11x [13] detection model without gradient flow.
- Minimum valid detection confidence of 0.1
- Minimum valid detection bounding box IoU of 0.5
- Bin size of 0, so we perform naive rasterisation.