Improving Perception via Sensor Placement: Designing Multi-LiDAR Systems for Autonomous Vehicles

Sharad Chitlangia[1*], Zuxin Liu[2*], Akhil Agnihotri[3], Ding Zhao[2]
[1]BITS Pilani, [2]CMU, [3]J. P. Morgan Chase and Co.
f20170472@goa.bits-pilani.ac.in, zuxinl@andrew.cmu.edu, agnihotri.akhil@gmail.com, dingzhao@cmu.edu

Abstract

Recent years have witnessed an increasing interest in improving the perception performance of LiDARs on autonomous vehicles. While most of the existing works focus on developing novel model architectures to process point cloud data, we study the problem from an optimal sensing perspective. To this end, together with a fast evaluation function based on ray tracing within the perception region of a LiDAR configuration, we propose an easy-to-compute information-theoretic surrogate cost metric based on Probabilistic Occupancy Grids (POG) to optimize LiDAR placement for maximal sensing. We show a correlation between our surrogate function and common object detection performance metrics. We demonstrate the efficacy of our approach by verifying our results in a robust and reproducible data collection and extraction framework based on the CARLA simulator. Our results confirm that sensor placement is an important factor in 3D point cloud-based object detection and could lead to a variation of performance by $10\% \sim 20\%$ on the state-of-the-art perception algorithms. We believe that this is one of the first studies to use LiDAR placement to improve the performance of perception.

1. Introduction

LiDAR sensors are becoming the critical 3D sensors for autonomous vehicle (AV) since they could provide accurate 3D geometry information and precise distance measures under various driving conditions. The point cloud data generated from LiDARs has been used to perform a series of perception tasks, such as object detection and tracking [32, 31]. Unlike the RGB image data from camera sensors, point cloud data is irregular and sparse, which brings great challenges for the 3D object detection task.

High-quality point cloud data and effective perception algorithms are usually both required to achieve accurate 3D object detection in practice. A number of studies propose to improve the 3D object detection performance for point cloud data by modifying the underlying perception algorithm by assuming that the data is rich and of high quality. This is usually done on public datasets using novel model architectures and effective data processing methods [32, 31]. However, there is only sparse literature considering the perception and detection problem from a data acquisition or LiDAR sensing point of view [23, 18]. We believe that this new perspective should be equally important for real-world AV applications since poor-quality sensing data could easily corrupt the perception algorithm and lead to poor performance [8, 22]. In addition, as suggested by Liu *et al.* [23], different LiDAR configurations will produce different point cloud distributions, which may affect the performance of downstream perception tasks. Therefore, we aim to investigate the interplay between LiDAR sensor placement and perception performance for AVs. We shall use placement and configuration interchangeably throughout this paper.

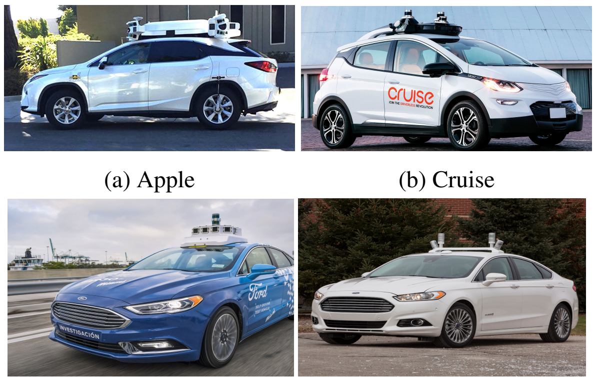

(a) Apple (b) Cruise
(c) Ford (d) UM PERL Lab

Figure 1: LiDAR configurations used in different autonomous vehicles [1, 3, 4, 2, 18].

We are interested in this topic for several reasons. On the one hand, given a specific perception algorithm, we want to evaluate the effectiveness of different LiDAR placement layouts based on the perception performance. On the other

*equal contribution

hand, the LiDAR placement may affect the sensing procedure and the point cloud input for the perception algorithm, which influences the perception quality [18, 23, 26]. Moreover, LiDAR is an expensive sensor, so it would be beneficial to minimize the LiDAR sensor deployment cost while maintaining the perception performance, which could be done by sensor placement optimization. As shown in Figure 1, many companies' self-driving cars are equipped with more than 2 LiDARs, which may provide more effective perception but the deployment cost would be high. Decreasing the number of LiDAR sensors or the number of laser beams may affect the perception system, but how to quantify the loss of performance and could we mitigate this loss by optimal LiDAR placement are questions that are rarely studied in the literature. Therefore, thoroughly studying the interplay between LiDAR sensor placement and perception performance is an essential bridge to obtain the right balance between the perception performance and affordability of AV perception systems, without sacrificing driving safety.

In this paper, we study the perception system from the sensing perspective. Particularly, we focus on investigating the relation between LiDAR sensor placement and 3D object detection performance. The contributions of this paper are summarized as follows:

1. As far as we are aware, we are the first work to quantitatively study how LiDAR placement affects the 3D object detection performance in a realistic simulation environment.

2. We propose an information-theoretic cost metric to optimize the LiDAR sensor placement, and we show that the surrogate metric is highly correlated with 3D object detection performance. The surrogate cost is easy to compute and could greatly accelerate the LiDAR sensor placement optimization procedure.

3. We contribute an automated multi-LiDAR data collection framework, which is heavily grounded in realistic traffic scenarios for strong reproducibility. We conduct extensive experiments based on the framework in CARLA simulator [12] and provide insightful application examples which could aid in more effective deployment of LiDAR sensors on AVs.

The remainder of the paper is structured as follows: Section 2 provides a brief overview of related work and suggests the research gap in the existing literature. Section 3 starts by formulation of the sensor placement optimization problem and introduces the surrogate cost function. We present the simulation environment, data collection and evaluation along with comparative results in Section 4. Then, Section 5 lists some promising applications of our framework. Finally, Section 6 concludes with a summary of our contributions and directions for future research.

2. Related Work

The methods and frameworks proposed in this work revolve around optimizing the LiDAR sensor placement to maximize point cloud based 3D object detection performance surrounding an AV. Although literature is scarce in this combined area, there has been research on the 3D object detection and the LiDAR placement optimization topics independently, which we discuss in this section.

3D object detection with point clouds. To keep up with the surge in interest in autonomous vehicles, researchers have tried to develop novel object detection, and more generally, perception algorithms for efficient point cloud-based 3D object detection. Majority of the previous methods developed use carefully crafted design which rely on the assumption of rich and complete point cloud data and do not take into account the variability of the underlying sensing system [20, 25, 5, 28].

Over the years, there has also been progress in grid-based and point-based detection methods for multi-sensor point cloud data. Grid-based methods project the point cloud data onto a plane for 2D treatment, and provide strategies for multi-sensor data fusion [11, 19, 21]. However, these methods are greatly limited by the kernel size of the convolution. To overcome this, a highly ubiquitous grid-based sliding window approach has also been studied with respect to point cloud detection [34, 13]. It relies on a popular voting scheme which accelerates the exhaustive 3D window searching. However, it is only efficient for sparse data points, similar to the framework by Su *et al.* [33], and couldnot be scaled to a dense point cloud, as observed in majority of AV applications.

Point-based methods, on the other hand, directly apply on the raw point cloud, rather than on the converted 2D images. This enables them to have flexible perception leading to robust point cloud learning [27, 35]. Generally speaking, these point based methods have higher receptive fields and higher computation costs than grid based methods. Therefore, to take advantage of lower computation of grid based methods and higher sensing capabilities of point based methods, Shi *et al.* [30] propose a robust framework, PointVoxel-RCNN (PV-RCNN), to unify the two methods. PV-RCNN utilizes a mixture of voxel Convolutional Neural Network (CNN) and PointNet-based set abstraction to efficiently learn features from raw point cloud data. Compared to conventional point-based methods, this framework provides for richer context information leading to a more accurate estimation of objects' positions and superior performance on the standard KITTI dataset [17]. More details are provided in Section 4.

LiDAR placement for autonomous vehicles. One of the most important factors critical to AV deployment is its perception and sensing ability. With this respect, LiDARs have been widely used because of their high real-time precision and their ability to extract extensive information from their environment [24, 17, 37]. Since, for AVs, the perception ability of a LiDAR is sensitive to its placement [36, 14], it is critical to develop a scheme that minimizes the uncertainty among all its possible placements. To this extent, Dybedal *et al*. [15] proposed to find the optimal placement of 3D sensors using a Mixed Integer Linear Programming approach, which is not scalable to AVs because of large number of variables involved. Rahimian *et al*. [29] develop a dynamic occlusion-based optimal placement routine for 3D motion capture systems, but do not take into account variable number of sensors during its optimization routine.

There have also been some prominent advances to optimize the placement of multiple LiDARs for AV while considering the perception performance. Mou *et al*. [26] formulate a min-max optimization problem for LiDAR placement with a cylinder-based cost function proxy to consider the worst non-detectable inscribed spheres formed by the intersection of different laser beams. However, their Mixed Integer Linear Programming solver could hardly be applied to a large number of LiDARs or laser beams because of its exponential computational complexity, and moreover, their optimization precision is limited. Liu *et al*. [23] improves the previous work by using an intuitive volume to surface area (VSR) ratio metric and a black-box heuristic-based optimization method to find the optimal LiDAR placement. While their approach aims to minimize the maximum non-detectable area, however, they assume a uniformly weighted region around the AV. In addition, they do not explicitly reveal the relation between the LiDAR placement and the perception performance, which could be done in realistic simulation environment or by real-world testing.

Our work overcomes these limitations of previous LiDAR placement works by utilizing a data-driven surrogate cost function with a ray-tracing acceleration approach. Since different cities may have totally different road infrastructure layouts and traffic patterns, we aim to optimize the LiDAR configurations accordingly by analysing collected data and maximizing the information gain so that the overall perception capability could be improved. Furthermore, we combine the two fields - 3D object detection and LiDAR placement - together and propose a systematic framework to quantitatively describe the relation between LiDAR sensing and perception. More details are provided in Section 3.

3. Method

In this section we introduce the LiDAR sensor and its perception areas. The problem of optimal LiDAR placement is formalized and related definitions of region of interest (ROI) and Probabilistic Occupancy Grid (POG) are presented. Finally, we describe how to compute the surrogate cost for LiDAR configuration optimization.

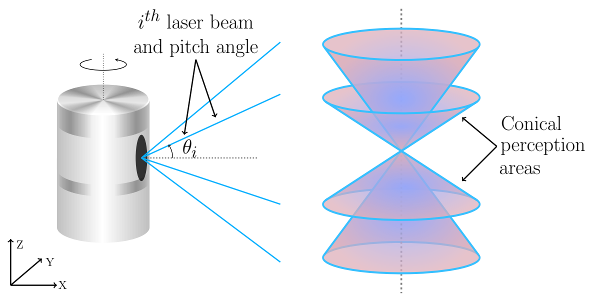
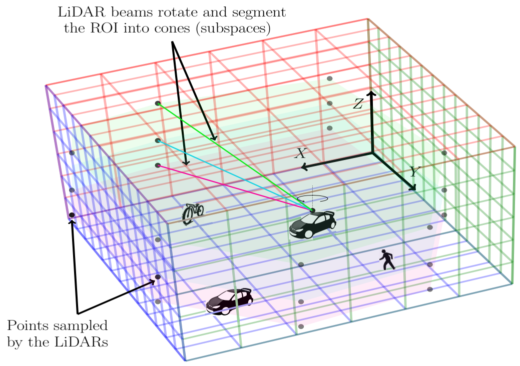


Figure 2: Schematic showing a LiDAR sensor forming perception cones in the ROI to collect point cloud data.



LiDAR beams rotate and segment
the ROI into cones (subspaces)
Z
X
Y
Points sampled
by the LiDARs

Figure 3: LiDAR sensor mounted on an AV samples points from the ROI and generates a point cloud transformed to the AV's coordinate system. These points along with their corresponding 3D bounding boxes generate the POG.

3.1. Problem formulation

We begin this section by defining the LiDAR perception model and the ROI, both of which form the basis of our optimal configuration problem.

As shown in Figure 2, we model a LiDAR sensor as a collection of multiple beams. Each beam makes a pitch angle with the XY plane and rotates with a uniform speed having the direction of rotation along the positive Z axis. As each beam completes one rotation, it forms a conical surface (area) and we assume its perception to be all points in this area. Thus, total perception of a LiDAR is the union of all these conical areas formed by rotation of its beams in the ROI, which we describe next.

We define our ROI to be the space where we keep track of objects to be detected. To account for LiDAR's limited range of detection, we fix the cuboid dimensions of the ROI to be [60,20,4] meters in the XYZ coordinate system throughout the paper, as shown in Figure 3. The ROI width (Y axis) is shorter than length (X axis) because an AV's longitude velocity is usually lesser than its latitude

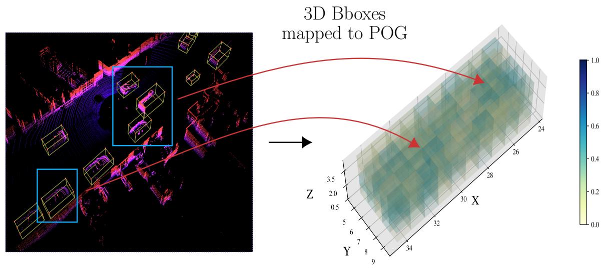


Figure 4: Schematic showing how 3D bounding boxes (Bboxes) are mapped to the ROI to generate a POG. Each cube in the POG has a probability given by Equation 1.

velocity and thus we want the AV to see further along X axis. We then discretize ROI into cubes with a fixed resolution, and represent the ROI as a collection of cubes. For instance, we use cubes of side 0.05 meters in the experiments, which results in a collection of $\frac{60}{0.05} \times \frac{20}{0.05} \times \frac{4}{0.05} = 38400000$ cubes to represent our ROI, which is denoted by $\{c_1, c_2, ..., c_{38400000}\}$. The ROI provides us with a fixed perception domain around the LiDAR, from which LiDAR beams accumulate maximum information. Similar LiDAR sensor modeling and ROI definition could also be found in Liu *et al*. [23] and Mou *et al*. [26].

Then, the problem of optimal LiDAR placement is formulated as finding a LiDAR configuration on the AV such that the cost function is minimized in the ROI. To evaluate the perception performance of a LiDAR placement, three commonly used metrics in 3-D Object Detection — Bird-eye view (BEV) detection, 3-D Intersection Over Union and Orientation Similarity — are used. The details of those metrics could be found in Kitti dataset [17]. However, directly using those metrics to optimize the LiDAR placement is extremely inefficient, as each optimization iteration may take days to collect and annotate new data based on new LiDAR placement, train detection models based on the data, and finally get the metrics. Therefore, we propose a surrogate cost function to accelerate the LiDAR placement optimization procedure.

3.2. Surrogate cost function

To maximize perception capability, an intuitive measure is to collect as much information as possible given a budget of LiDARs. While Liu *et al*. used the notion of maximum non-detectable subspaces to reflect perception performance, we propose a data-driven measure to quantify the information gain. A key insight here is that AVs are deployed in different cities, so they encounter varied conditions including but not limited to road infrastructures and traffic patterns. Therefore, we aim to optimize LiDAR placement such that the lasers could focus on important areas around the AV based on customized datasets of different cities or scenarios.

To this end, we propose to model the ROI, which is represented by a collection of cubes, as a Probabilistic Occupancy Grid (POG) by calculating the probability of each cube to be occupied. This grid is created by registering information about the 3D bounding box (Bbox) positions of objects of interest, such as cars and pedestrians. We will use car as an example throughout the paper. Suppose we have a Bbox dataset $\mathcal{D}_T = \{d_1, d_2, ..., d_T\}$, where T represents the total size. Each frame $d_t, (t \in \{1, ..., T\})$ contains N 3D Bboxes of cars $\{b_{t1}, ..., b_{tN}\}$ within ROI of the ego-vehicle, and each Bbox is parameterized by its center coordinates, size (length, width, height), and yaw orientation. Given a coordinate $c = (c_x, c_y, c_y)$, we denote $c \in d_t$ if c is within any of the bounding boxes $\{b_{t1}, ..., b_{tN}\}$. Then, for each cube c_i in ROI, we could calculate its probability to be occupied as:

$$P(c_i) = \frac{\sum_{j=1}^{T} \mathbb{I}(c_i \in d_j)}{T} \tag{1}$$

where $\mathbb{I}(\cdot)$ is an indicator function. The POG could then be formally defined as $\{P(c_1), P(c_2), ..., P(c_M)\}$, a collection of occupancy probabilities of each cube in the ROI, where M is the total number of cubes in ROI. One such example of a POG is shown in Figure 4.

For a particular LiDAR configuration C, we employ ray tracing by Bresenham's Line Algorithm [7] to find all the cubes which intersect with the perception areas of that LiDAR. We denote the set of these perception cubes as $S_C = \{s_1, s_2, ..., s_n\}$, where n is the set size. We could consider this collection of cubes to be discretized trajectories of all LiDAR beams and to represent the perception range of this LiDAR configuration C.

Given the set of perception cubes S_C, the probability of each cube signifies presence of objects inside it. Since presence of an object in one cube does not imply presence in other cubes, we could treat these probabilities independently and calculate the joint distribution over all cubes in the set S_C as:

$$P(S_C) = P(s_1, s_2, ..., s_n) = \prod_{k=1}^{n} P(s_k) \tag{2}$$

Considering the probabilistic nature of the problem, information about presence of objects in a cube could be quantified using Shannon Entropy of the probability distribution defined over the cube.

$$H(s_i) = -p\log(p) - (1-p)\log(1-p)\ ;\ p = P(s_i) \tag{3}$$

The uncertainty information of a particular LiDAR configuration C could then be quantified as:

$$H(S_C) = H(s_1, s_2, ..., s_n) = \sum_{i=1}^{n} H(s_i) \tag{4}$$

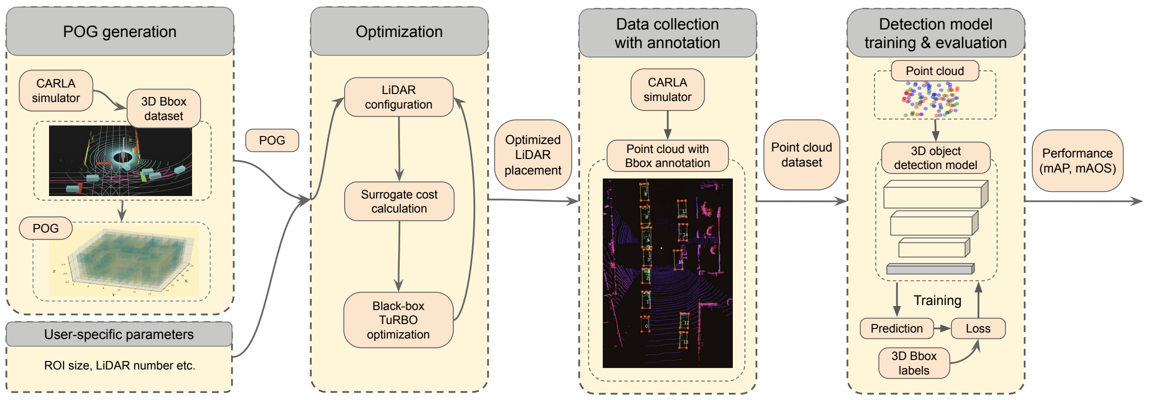

POG generation
CARLA simulator
3D Bbox dataset
POG
User-specific parameters
ROI size, LiDAR number etc.
POG
Optimization
LiDAR configuration
Surrogate cost calculation
Black-box TuRBO optimization
Optimized LiDAR placement
Data collection with annotation
CARLA simulator
Point cloud with Bbox annotation
Point cloud dataset
Detection model training & evaluation
Point cloud
3D object detection model
Training
Prediction
Loss
3D Bbox labels
Performance (mAP, mAOS)

Figure 5: Evaluation framework overview.

Finally, we define the surrogate cost of a LiDAR placement C to be $cost(C) = -H(S_C)$ and we aim to minimize this cost. This is equivalent to maximizing the amount of information about presence of objects in the cubes of set S_C covered by the LiDAR configuration C.

3.3. Optimization

A single LiDAR configuration could be expressed as $[x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i]$, where (x_i, y_i, z_i) is the location and $(\alpha_i, \beta_i, \gamma_i)$ are roll, pitch and yaw angles of the LiDAR in the ROI coordinate system. The yaw angle (represented by γ_i) is not optimized as the LiDAR beams rotate 360 degrees in the horizontal FOV. Therefore, our goal is to find a configuration $[x_i, y_i, z_i, \alpha_i, \beta_i]$ which minimizes the uncertainty mentioned in Equation 4.

Since the surrogate cost metric is not differentiable, we could use any black-box optimization method to find the optimal LiDAR configuration. A proposed configuration C is considered to be valid if it satisfies some geometric constraints, because the LiDARs could not be installed far away from the vehicle. These geometric constraints are also the boundary points of the search region. Denote $C_{min} = [x_{min}, y_{min}, z_{min}, \alpha_{min}, \beta_{min}], C_{max} = [x_{max}, y_{max}, z_{max}, \alpha_{max}, \beta_{max}]$. Then, the optimization objective becomes:

$$\begin{aligned} C^* &= \arg\min_{C} -H(S_C) \\ &\text{such that} \quad C_{\min} \leq C \leq C_{\max} \end{aligned} \tag{5}$$

The overall pipeline of our LiDAR configuration optimization and evaluation method is shown in Figure 5. We utilize TuRBO [16] as the optimizer to minimize the surrogate. TuRBO works by creating multiple local probabilistic models instead of a single global exploration model. Similar to stochastic optimization methods, it creates trust regions (the center of which is usually the best solution) using simple surrogate models that are *believed* to accurately model the function to be optimized. TuRBO proposes configurations which are then evaluated using our ray tracing based fast evaluation methodology on top of a POG.

4. Experiments

In this section, we aim to address two questions: 1) Does LiDAR placement influence the final perception performance? 2) Could our LiDAR placement optimization method improve the perception performance? To answer these questions we conduct extensive experiments in a realistic self-driving simulator — CARLA [12]. We choose to evaluate our method in simulation environment rather than on public datasets or in real world for several reasons. First of all, public point cloud datasets are collected from specific hardware setups, which could not generate various LiDAR placement configurations as required. Secondly, we need to fix all environment variables, such as the ego-vehicle trajectory and surrounding objects, and only change the LiDAR configurations to fairly compare the perception performances of these configurations. However, it is very hard to control these environment variables for different LiDAR configurations in the real world because vehicles and pedestrians on the road are continuously changing. Therefore, the most convenient and economical way to achieve this is to simulate scenarios and point clouds in realistic simulators such as CARLA.

In the subsequent subsections we detail the experimental setup in CARLA and show how it connects to the overall pipeline of our methodology, as shown in Figure 5.

4.1. Data collection and model training

CARLA simulator. We use CARLA as the simulation platform to collect data and evaluate our method. CARLA is a high-definition open-source simulator for autonomous driving research that offers flexible scenario setups and sensor configurations. CARLA provides realistic rendering and physics simulation based on the Unreal Engine 4, and

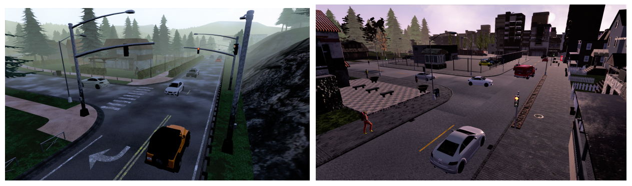


Figure 6: Sample maps in the CARLA simulator used for data collection. (a) An infinite map loop with a highway and a small town. (b) A small town with basic road junctions.

has a growing community to support it and develop new features. Furthermore, the LiDAR sensor could generate precise point cloud data and could be easily configured with customized placement and beam parameters. Hence, CARLA has been used for a wide range of AV research, including development, training, testing and validating autonomous driving systems [9, 10].

ScenarioRunner. The CARLA community contributes many urban maps and scenarios. For example, Figure 6 shows a sample of the maps provided in the simulator, which encompass various conditions of traffic and road geometries. To fairly compare perception capabilities of different LiDAR configurations we use CARLA's inbuilt *ScenarioRunner* module to simulate traffic scenarios such that all environmental variables, except for the LiDARs, are the same. A scenario is usually composed of 3D models of static and dynamic objects, where the former could be regarded as the overall map including roads and buildings, and the latter as pedestrians and vehicles. The perception algorithm that is evaluated in this paper aims to detect dynamic objects and predict their 3D bounding boxes. The ego-vehicle with LiDAR sensors and other dynamic objects keep the same moving trajectories in these scenarios. This ensures that the ground-truth 3D bounding boxes for all dynamic objects around the ego-vehicle within the ROI are the same for different LiDAR configurations. We collect data from 87 such scenarios with fixed 10 hz sampling frequency, and the overall dataset size is 35000 frames, which contains point clouds and 3D bounding box labels for dynamic objects. The point clouds from different LiDARs are transformed to the ego-vehicle's frame of reference for computational convenience.

Data format. The data collection procedure is tied as close as possible to the KITTI dataset [17]. Using *ScenarioRunner* ensures strong reproducibility of experiments. To track the visibility of objects, a depth camera is utilized. 3-D Bounding boxes are projected back to the camera image using intrinsic and extrinsic matrices for tracking the height and truncation of the object in the bird's eye view. Using occlusion statistics, height and truncation, objects are classified as easy or hard similar to the KITTI dataset.

3D object detection model. We use PointVoxel-RCNN (PV-RCNN) [30] as the 3D object detection module to evaluate the perception performance of different LiDAR placements. PointVoxel-RCNN combines voxel-based features using anchors of various different sizes and and pointnet-based features using a Voxel Set Abstraction Layer. Following this layer, using multiple receptive fields, the learned discriminative features of keypoints are then aggregated to the ROI-grid points for finer grained region proposals. The training hyper-parameters, such as the number of iterations, learning rate, and batch size, are kept the same for all experiments for fair comparison. See the supplementary material for details.

Baselines. We adopt several baseline LiDAR placements to show the effectiveness of our proposed optimization method. We consider the 4 LiDAR placement problem where each LiDAR has 16 beams. Although the beam angles could be optimized (see Section 5.4 for details), for our experiments we consider beams to be equally spaced in the vertical FOV $[-25.0, 5.0]$. The first baseline configuration is inspired by Apple's self-driving cars (Figure 1a), which places the 4 LiDARs on each of the 4 roof corners. We name this placement as "**Square**". The second baseline configuration is achieved by stacking four LiDARs together on the center of the roof, which is inspired by Ford's autonomous vehicle (Figure 1c) and we name it as "**Center**". The third baseline is motivated by Cruise (Figure 1b). The 4 LiDARs are placed in a line, where the 2 LiDARs on both sides are tilted at a certain angle. We name it as "**Line**". The visualization of baseline placements are shown in the first column of Figure 7, where each LiDAR's pose is with respect to ego-vehicle's coordinate system having origin at the vehicle's geometric center. The detailed coordinates of each LiDAR placement are presented in the supplement.

Evaluation metrics. We show results on three commonly used metrics — Bird-eye view (BEV) detection, 3-D Intersection Over Union (IOU) and Average Orientation Similarity (AOS) — to evaluate the final perception performance of different LiDAR configurations. 3-D IOU measures the fraction of overlap region divided by total region occupied by the 3-D bounding boxes together. Orientation Similarity considers prediction of the orientation together with object detection. These metrics are reported by averaging precision across 40 recall points so as to roughly approximate the precision recall curve. The evaluation IoU thresholds are set as 0.7 and 0.5 for easy and hard respectively. The detail of the metrics is described in [17]. As a particular instance of this paper, we only evaluate the detection performance for the 'Cars' label.

Note that all the environmental variables (scenarios and model training parameters) except the LiDAR configurations **are always the same** during the perception performance evaluation procedure, so the three metrics could reflect the point cloud quality or the perception capability for

Configuration	Car - BEV			Car - 3D Detection			Car - AOS		
	AP_R40	@0.70	@0.50	AP_R40	@0.70	@0.50	AP_R40	@0.70	@0.50
	Overall	Easy	Hard	Overall	Easy	Hard	Overall	Easy	Hard
Square	*50.54*	60.11	40.97	*46.81*	55.48	38.14	*35.6*	39.57	31.76
Center	*49.01*	56.33	41.69	*43.94*	50.75	37.12	*49.83*	53.14	46.51
Line	*45.18*	51.68	38.68	*42.79*	49.25	36.32	*31.02*	33.98	28.07
Optimized (Ours)	***55.47***	**60.78**	**50.16**	***50.53***	**56.82**	**44.25**	***53.48***	**58.96**	**47.99**

Table 1: Comparison of object detection performance of various LiDAR configurations. For all 3 metrics, **mean average precision** is computed by averaging precision across 40 recall positions (**AP_R40**). **Overall** column is the arithmetic mean of **Easy** and **Hard** columns. More details about their definitions could be found in the KITTI dataset [17]. The IOU thresholds in the evaluation are set as 0.7 and 0.5 for easy and hard respectively.

different LiDAR placements.

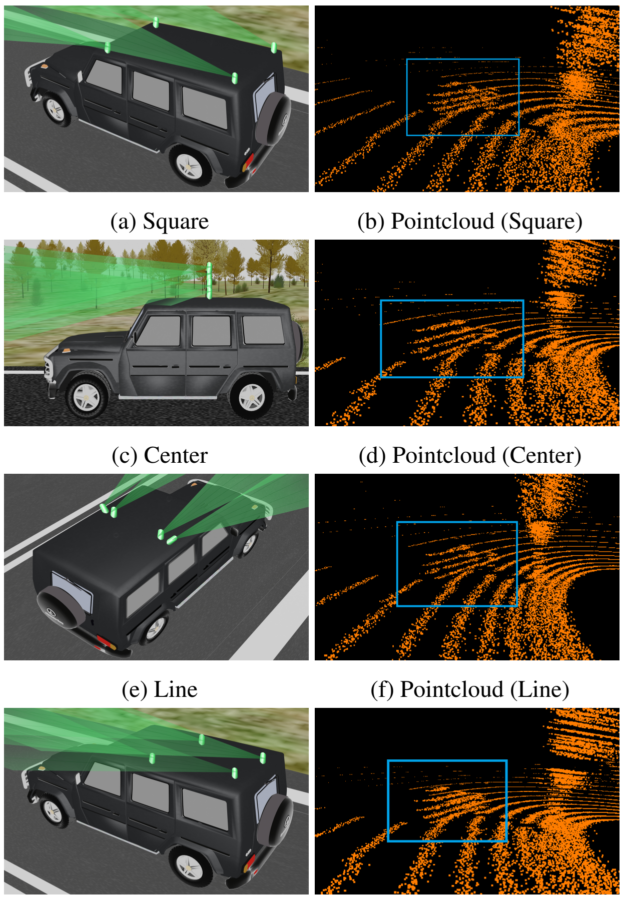

(a) Square (b) Pointcloud (Square)
(c) Center (d) Pointcloud (Center)
(e) Line (f) Pointcloud (Line)
(g) Optimized (h) Pointcloud (Optimized)

Figure 7: Standard baseline and optimal LiDAR configurations along with their generated point clouds, used for Experiment 1. The left column above shows the configurations while the right column displays the corresponding point clouds in the same scene.

4.2. Results

The results are presented in Table 1, which clearly show that different LiDAR configurations will affect the perception performance because all other variables (number and type of LiDAR, environment, object trajectories, model training parameters, etc) are the same except the LiDAR placement. It is seen that the '**Square**' configuration, which is inspired by Apple's AV, outperforms other standards baselines in consideration.

Furthermore, across the various baseline configurations and evaluation metrics, we find that models trained with optimal placement data outperform models trained on baseline configurations. We could see a large performance improvement with the optimized placement when compared with baselines ($10\% \sim 20\%$), which implies that our LiDAR placement optimization method could provide better point cloud quality and thus improve the perception performance. Figure 8 shows the relation of our surrogate cost value with one of the perception evaluation metric IOU. We could see that the proposed information-theoretic cost is inversely correlated with the evaluation metric, which means that minimizing the surrogate cost could improve the perception capability. The second column in Figure 7 shows the point clouds of a certain scenario with different LiDAR configurations. We can see that the optimal placement leads to denser points on the car.

5. Applications

In this section we detail some of the applications and extensions of our methodology. We believe that the following subsections have direct relevance for AV researchers and manufacturers. We provide more experiments for some applications in the supplementary material.

5.1. Accelerated LiDAR placement evaluation

Since our surrogate cost is inversely correlated with the perception performance, we could accelerate the evaluation procedure of a particular LiDAR configuration in a city. In

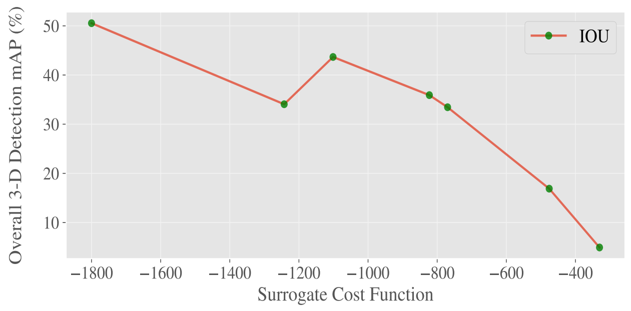


Figure 8: Relation of overall mAP values of IOU (in %) and the surrogate cost function.

other words, using our surrogate cost, one does not need to follow the time-consuming procedure to get an evaluation of a LiDAR placement: LiDAR installation $\rightarrow$ massive data collection $\rightarrow$ perception model training $\rightarrow$ evaluation of perception performance. Instead, one only needs a 3D bounding box dataset of object of interests to generate the POG and evaluate the LiDAR placement, which is fast and economical, and could also be customized by different deployment scenarios.

5.2. LiDAR placement optimization

Since we could evaluate a LiDAR placement fast using our surrogate cost function in Equation 4, it is now easy to optimize a LiDAR setup given the number of LiDARs (and their beams) in the setup. From Table 1, we could also see a great performance improvement by our optimized LiDAR placement. This has direct relevance to the AV research community and industry since the current LiDAR placement is more or less intuition driven. Out approach could maximize the efficacy of the LiDAR sensor usage.

5.3. LiDAR number selection

With the proposed framework, we could easily evaluate the perception performance as the number of LiDARs increase and choose the desired one. For instance, given a type of LiDAR, we could find the optimized placement for n LiDARs, where $n = \{1, 2, 3, ...\}$, and compute the surrogate cost easily based on a POG. Then we plot the surrogate cost value versus the number of LiDARs and pick the elbow point number as the optimal choice, because after this point there would be marginal performance improvement by increasing the LiDAR number. Therefore, our method could help the AV manufactures choose the right balance between the sensor cost and perception capability and avoid unnecessary expenditure.

5.4. LiDAR beam angle optimization

Since the beam angles of a LiDAR are also tunable, a logical follow-up question which arises following subsection 5.2 is finding the beam angle which maximizes object detection. Specifically, we could regard a multi-beam LiDAR as a collection of multiple single beam LiDARs at the same position, then use our method to solve optimal beam angle design for the LiDARs such that the information gain is maximized. Hence, our method could also help the LiDAR manufactures design their sensor specifications based on different AV deployment scenarios.

5.5. Active LiDAR perception

AVs with LiDAR sensors have to process a large amount of point cloud data online, which brings a lot of computational burden and is not energy efficient. Therefore, some active depth sensors have been proposed to sense the environment dynamically, such as the light curtain sensor developed by Bartels *et al.* [6]. The proposed surrogate cost in this paper could serve as a guidance for those active sensors to focus on important areas around the AV based on different scenarios.

6. Conclusion

This paper investigates the interplay between LiDAR placement and perception performance. We proposed a novel solution to the problem of optimal LiDAR placement and configuration for AVs. Our motivation stemmed from building a unified framework of optimal LiDAR perception and configuration, which will help balance perception capabilities and design costs of LiDARs, without sacrificing safety in AVs. We proposed a data-driven surrogate cost function which characterized the total uncertainty of information present in the conical perception areas. This helped us accelerate LiDAR placement optimization procedure that aims to maximize LiDAR perception performance. Finally, we conducted extensive experiments in an AV simulator, CARLA, to validate the correlation between LiDAR perception and configuration. We employed the state-of-the-art 3D object detection module to evaluate and compare different LiDAR configurations. The experimental results indicated that the optimal LiDAR placement solved by our method outperforms those of standard baseline configurations.

Research in this paper sets the precedent for future work which will consider occlusion between the LiDARs and the ego vehicle. With increasing relevance of adversarial learning, we also aim to investigate if we could mitigate the effect of sensor attacks on the AV perception by changing the LiDAR placement.

References

[1] Apple autonomous vehicle. https://www.teslarati.com/apple-car-larger-fleet-new-hires. Accessed: 2021-03-10.

[2] Cruise autonomous vehicle. https://www.wired.com/story/gm-cruise-generation-3-self-driving-car. Accessed: 2021-03-10.

[3] Ford autonomous vehicle. https://www.theverge.com/2018/8/16/17693866/ford-self-driving-car-safety-report-dot. Accessed: 2021-03-10.

[4] University of michigan autonomous vehicle. http://robots.engin.umich.edu/Projects/NGV. Accessed: 2021-03-10.

[5] A. Agnihotri, P. Saraf, and K. R. Bapnad. A convolutional neural network approach towards self-driving cars. In *2019 IEEE 16th India Council International Conference (INDICON)*, pages 1–4, 2019.

[6] Joseph R Bartels, Jian Wang, William Whittaker, Srinivasa G Narasimhan, et al. Agile depth sensing using triangulation light curtains. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7900–7908, 2019.

[7] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[8] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019.

[9] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[10] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2765–2771. IEEE, 2019.

[11] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[13] Hongyuan Du, Linjun Li, Bo Liu, and Nuno Vasconcelos. Spot: Selective point cloud voting for better proposal in point cloud object detection. In *European Conference on Computer Vision*, pages 230–247. Springer, 2020.

[14] Hugh F Durrant-Whyte. Consistent integration and propagation of disparate sensor observations. *The International journal of robotics research*, 6(3):3–24, 1987.

[15] Joacim Dybedal and Geir Hovland. Optimal placement of 3d sensors considering range and field of view. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1588–1593. IEEE, 2017.

[16] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 5496–5507, 2019.

[17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[18] Rohit Ravindranath Kini. Sensor position optimization for multiple lidars in autonomous vehicles, 2020.

[19] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[20] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

[21] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[22] Daniel Liu, Ronald Yu, and Hao Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283. IEEE, 2019.

[23] Zuxin Liu, Mansur Arief, and Ding Zhao. Where should we place lidars on the autonomous vehicle?-an optimal design approach. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2793–2799. IEEE, 2019.

[24] Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*, volume 2, page 3, 2014.

[25] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[26] Shenyu Mou, Yan Chang, Wenshuo Wang, and Ding Zhao. An optimal lidar configuration approach for self-driving cars. *arXiv preprint arXiv:1805.07843*, 2018.

[27] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[29] Pooya Rahimian and Joseph K Kearney. Optimal camera placement for motion capture systems. *IEEE transactions on visualization and computer graphics*, 23(3):1209–1221, 2016.

[30] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[31] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[32] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[33] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018.

[34] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607. Rome, Italy, 2015.

[35] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[36] Hong Zhang. Two-dimensional optimal sensor placement. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(5):781–792, 1995.

[37] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

A. Appendix

A.1. Hyper-parameters

We use PV-RCNN as the primary 3D object detection model for our test bench. Details of the training hyper-parameters are given in Table 2. We ensure that the hyper-parameters are same for all experiments to fairly compare the point cloud quality.

Hyperparameter	Value
Epochs	30
Optimizer	Adam
Learning Rate	0.01
Weight Decay:	0.01
Momentum:	0.9
Learning Rate Clip	0.0000001
Learning Rate Decay	0.1
Div Factor	10
Warmup Epochs	1
Learning Rate Warmup	False
Gradient Norm Clip	10
MOMS	[0.95, 0.85]
PCT_START	0.4
Decay Step List	[35, 45]

Table 2: Hypeparameters for PV-RCNN

A.2. Coordinates of LiDAR Configurations

Following the notation given in Section 3.3, the ego-vehicle has its coordinate frame at its geometric center at $[30, 10, 0, 0, 0]$ with respect to the ROI frame of reference. All LiDAR configurations, which are mentioned in Section 4.1 are shown in Figure 7, and their detailed coordinates are given in Table 3. These coordinates are with respect to the ego-vehicle's coordinate frame, as illustrated in Figure 9.

A.3. Sample Evaluation of LiDAR Configurations

This section provides an example to demonstrate accelerated LiDAR placement evaluation, which is described in Section 5.1. Since the proposed POG-based metric is correlated with perception performance, we can easily compare two LiDAR configurations by their surrogate costs. Note that the two LiDAR configurations could have different number of LiDARs and even different types of LiDARs.

For instance, consider an AV company which has two LiDAR configuration candidates A and B with the same price, where A is 4 10-beam LiDARs and B is 1 40-beam LiDAR. Since their number of beams is equal, it is hard to tell which one will perform better given the company's point cloud perception algorithm.

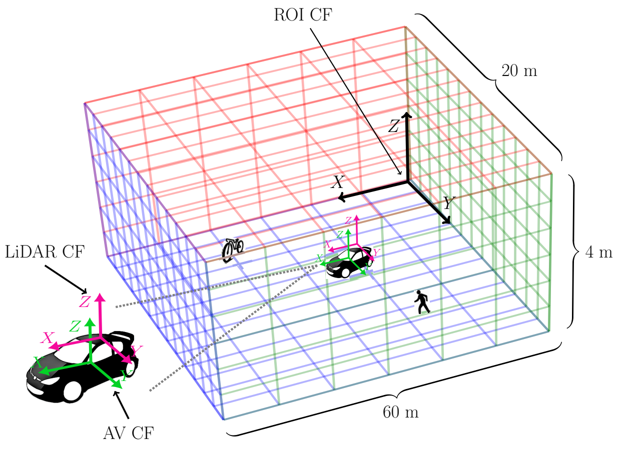


Figure 9: Coordinate frames (CF) of reference of ROI (origin at one of its corners), the ego-vehicle (origin at its geometric center) and a LiDAR. Figure not to scale.

One traditional way to evaluate the two candidates is to install those LiDARs on the vehicle, collect and annotate point cloud data, train the perception model, and finally obtain the perception performance. However, there are two major drawbacks for this approach: 1) Since we have showed that the sensor placement will influence the perception performance, how to install those LiDARs on the vehicle to fairly compare them? 2) The whole procedure requires a lot of time and labor, which introduces additional deployment costs.

In contrast, with our approach, we can easily optimize the placements for both A and B and compare their optimal surrogate costs. In this way, the evaluation procedure could be done in several hours. For example, we still consider the CARLA simulation environment and we have the same POG as we used in Section 4. Then we optimize the placements for A and B and get the optimal surrogate costs $cost(A) = -1383.685, cost(B) = -756.337$. Then, we know that A (4 10-beam LiDARs) is better then B (1 40-beam LiDAR) because $cost(A) < cost(B)$ with optimal placement, which signifies that A could gain more information than B.

A.4. How many LiDARs are enough?

This section provides an example for the LiDAR number selection application as described in Section 5.3. We are interested in this application for two main reasons: 1) To reduce deployment cost and real-time computational burden of point cloud data processing, which might be unnecessary in many cases. 2) To install sufficient number of LiDARs for safe and adequate perception of the environment for AV safety. Therefore, selecting the right number of LiDARs on AV to balance the cost and perception capability is extremely important.

	x	y	z	roll	pitch
Square	- 0.500	0.500	2.200	0.000	0.000
	- 0.500	- 0.500	2.200	0.000	0.000
	0.500	0.500	2.200	0.000	0.000
	0.500	- 0.500	2.200	0.000	0.000
Center	0.000	0.000	2.600	0.000	0.000
	0.000	0.000	2.600	0.000	0.000
	0.000	0.000	3.000	0.000	0.000
	0.000	0.000	3.000	0.000	0.000
Line	- 0.250	0.000	2.600	0.000	0.000
	0.250	0.000	2.600	0.000	0.000
	0.000	- 0.250	2.600	0.550	0.000
	0.000	0.250	2.600	2.700	0.000
Optimized	- 0.068	0.929	2.866	2.652	2.768
	- 0.412	- 0.277	2.432	0.404	2.886
	- 1.783	0.560	2.903	0.491	3.131
	- 1.867	- 0.534	2.926	0.411	2.850

Table 3: Coordinates of LiDAR sensors with respect to the ego-vehicle coordinate frame. The LiDAR coordinates are transformed from world (ROI) frame to the ego-vehicle frame of reference for computational convenience and intuitive understanding of their placement. All values are in meters.

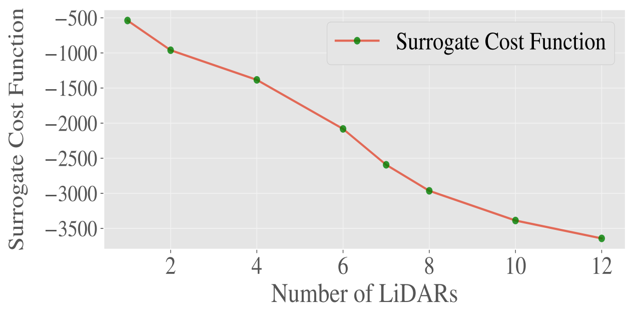


Figure 10: Relation of the surrogate cost function with the number of LiDARs.

As a particular example, suppose we have n 10-beam LiDARs for an AV and we want to determine the number n. Then we can compute the optimized placements for i LiDARs and obtain their corresponding surrogate cost $cost(i)$, where $i \in \{1, 2, ...\}$. Figure 10 shows the surrogate cost values versus the LiDAR number. We can see an elbow point at around $n = 7$, which means adding more LiDARs than 7 will not improve the perception performance a lot. Therefore, installing 7 10-beam LiDARs on an AV would be a good balance between the perception capability and deployment cost.