Deliverable D4.3

Back Loop Sensor Adaptation

Dissemination level	**PU**
Version	**1.0**
Lead contractor	**ALG**
Due date	**30.11.2022**
Version date	**09.12.2022**

Document information

Authors

Felix Heide – Algolux

Fahim Mannan – Algolux

Félix Goudreault – Algolux

Nicolas Robidoux – Algolux

Tom Riley – Algolux

Mario Bijelic – Algolux

Stefanie Walz – Mercedes-Benz

Dominik Scheuble – Mercedes-Benz

Funding

Co-labelled PENTA and EURIPIDES2 project endorsed by EUREKA, National Funding Authorities:
Austrian Research Promotion Agency (FFG)
Business Finland
Federal Ministry of Education and Research (BMBF)
National Research Council of Canada Industrial Research Assistance Program (NRC-IRAP)

Contact

Dr. Werner Ritter
Manager Vision Enhancement Technology
Environment Perception
Mercedes-Benz AG
RD/AFU
Werk/Plant 05919 - HPC G005-BB
D-71059 Sindelfingen, Germany
Mobile +49 (160) 863 8531
werner.r.ritter@mercedes-benz.com

LEGAL DISCLAIMER

Table of contents

List of figures

List of tables

1 Executive Summary

The scope of this report is limited to optimizing LiDAR (Light Detection and Ranging) signal processing and gated camera parameters. Optimization is primarily performed using synthetic data that is generated to replicate real-world scenarios. LiDAR processing parameters are optimized for 3D detection and depth estimation, and gating parameters are optimized for dense depth estimation.

1.1 LiDAR Signal Optimization

Scanning LiDAR systems are widely used for autonomous driving systems. LiDAR operates by emitting pulses of light and processing the returned waveform to determine the closest obstacle in the scene. Full waveform to single detection processing is performed by a black box digital signal processor (DSP) running a proprietary algorithm and hyperparameters are chosen independently of the downstream perception task or the operating domain. In contrast, we propose domain-specific black box hyperparameter optimization to improve 3D detection and depth estimation performance (“Black box”, here, means that knowledge of the internals of the LiDAR system, if available, are not used by the optimizer to compute, for example, derivatives. LiDAR pipeline algorithms are generally proprietary and LiDARs are consequently opaque). The main contributions of this work are, realistic simulation of full wave LiDAR using physically realistic Bidirectional Reflectance Distribution Functions (BRDF), Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) based black box optimization of the DSP hyperparameters, and joint optimization of LiDAR hyperparameters and perception tasks, namely, 3D detection and depth estimation. For depth estimation we achieve about 43% improvement in depth Root-Mean-Square Error (RMSE).

1.2 Gating Parameter Optimization

Gated cameras capture the depth-dependent intensity images of a scene. The quality of the captured image depends on the delay between illumination and exposure, number of laser pulses, gate duration and pulse duration. In prior captures, the parameters were manually chosen to cover a range of up to 150 m and are not optimized for depth quality. To obtain the best performance these parameters need to be optimized for the target depth estimation model. This is challenging as the gated imaging process is not differentiable. To this end, we fit a differentiable proxy to model the relationship between the gating parameters and the output gated image. This allows us to jointly optimize the gating parameters with the final depth estimation network. With this joint optimization approach, we are able to improve the depth quality by about 13% in terms of Mean Absolute Error (MAE) and about 6% in terms of RMSE.

2 Introduction

This report is a deliverable in the work package *WP4 Sensor Fusion and AI* titled "*Back loop sensor adaptation*". Sensors are a critical component of an autonomous system as they are the only interface to the outside world for the perception stack. Raw sensor data is typically processed by a DSP or an image signal processor (ISP) to increase the signal-to-noise ratio (SNR) before being passed to the perception stack. Conventional approaches optimize the raw data processing and perception task independently which can result in low perception performance. This report investigates back loop sensor adaptation which optimizes the sensor and processing parameters based on the output of the perception task resulting in robust perception under adverse weather conditions.

This report is presented in four chapters. Chapter 3 presents LiDAR signal optimization which optimizes the LiDAR's DSP parameters along with the 3D object detection and depth estimation models and chapter 4 presents gated camera parameter optimization which involves optimizing the gating parameters using a differentiable proxy to maximize the performance for dense depth estimation.

3 LiDAR Signal Optimization

LiDAR has become a cornerstone sensing modality for 3D vision in outdoor scenarios, especially as a safety-critical input to robotic decision-making algorithms in self-driving vehicles. LiDAR systems emit a pulse of light into the scene and then rely on low-level hardware DSP pipelines to construct a 3D point cloud (PC) from the wavefront of returned photons. Existing LiDAR DSPs contain cascades of parameterized operations and modifying these parameters may result in significant changes to the output PC and, de facto, to the downstream application ingesting it. Existing vision stacks aim to compensate for incorrect measurements in the downstream network. Departing from this approach, we investigate an end-to-end optimization approach that optimizes LiDAR DSPs along with the downstream network weights. To have ground truth data available, we developed a realistic LiDAR simulation environment based on the CARLA [1] engine that can generate raw and noisy waveforms which are fed to a DSP pipeline. We optimize the vision stack, including the DSP parameters and network weights, solving a nonlinear multi-objective optimization problem via a 0th-order stochastic algorithm both for 3D object detection Intersection-over-Union (IoU) losses and depth error metrics.

3.1 LiDAR DSPs in Simulation

Environment perception for autonomous drones [2], [3] and vehicles [4] requires precise depth sensing for safety-critical control decisions. Scanning LiDAR sensors have been broadly adopted in autonomous driving [5]–[7] as they provide high temporal and spatial resolution, and recent advances in Micro Electro-Mechanical Systems (MEMS) scanning [8] and photodiode technology [9] have drastically reduced their system cost and form factor.

Although existing 3D detection methods take 3D point cloud data as input, LiDAR sensors do not directly generate these point clouds but operate by sending out a laser pulse and measuring the temporal response through Avalanche Photo Diodes (APD). This temporal waveform signal is processed by applying a DSP step that extracts the peaks corresponding to echoes from possible targets within the scene [10]. As such, the DSP results in a data reduction of 1000× for a single emitted beam, producing a single or multiple 3D points per beam. Compressing the waveform into points in 3D space without too much loss of information is challenging in the presence of object discontinuities, sub-surface scattering, multipath reflections, or scattering media, see Figure 3.1. Scattering in particular, can occur in adverse weather conditions like fog [7], [11]–[16], rain [12], [16], [17] and snow [15], [18]–[20]. Today, LiDAR manufacturers handle all cases above by adjusting the DSP internal parameters in controlled environments and restricted real-world scenarios using a combination of visual inspection and depth quality metrics while, in a production environment, the LiDAR unit essentially becomes a black box where internal parameters remain hidden to the user.

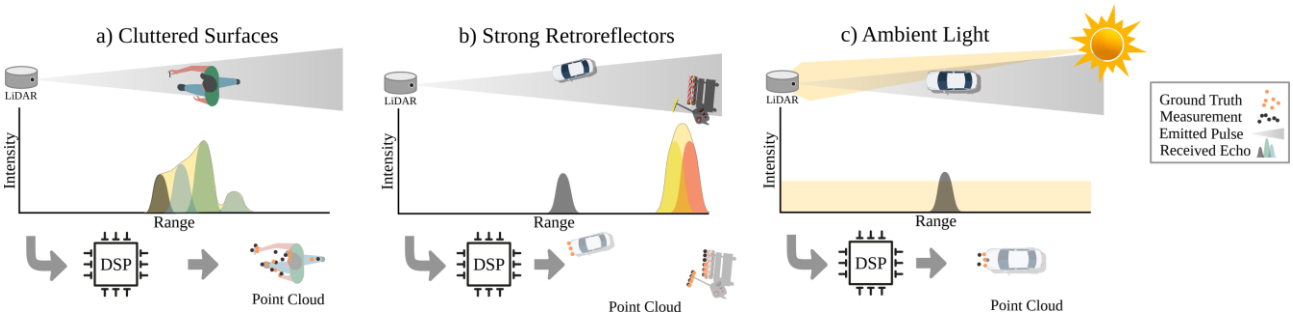

Figure 3.1: LiDAR Point Cloud Formation. Point cloud measurements generated by typical LiDAR sensors are produced by a multi-stage measurement and signal processing chain. The LiDAR emits a laser pulse. This signal travels through the scene and returns to the detector after single or multiple reflections. Cluttered surfaces (a), strong retro reflectors (b) and ambient light are introduced (c) in the returned signals. Thus, the full transient waveform read by sensors is the superposition of multiple return paths. The DSP, itself a chain of processing blocks, processes all temporal waveforms and extracts a constant stream of 3D points that forms the final point cloud (bottom).

To account for noisy and incorrect point cloud estimates with these black box DSPs, existing work has explored simulation methods that augment real datasets with simulated adverse effects and point cloud degradations for rain [17], [21], [22], fog [7], [13], [23] and snow [20]. With augmented point clouds in hand, downstream vision models are retrained to produce predictions robust to the point cloud data degradation. Another approach consists of generating synthetic data in 3D scenes using rendering engines [24]–[26]. These existing methods typically avoid simulating transient light propagation and signal processing by ray casting a virtual 3D scene directly into a point cloud, thus very roughly approximate point cloud noise. Moreover, simulation methods that generate synthetic point clouds or alter measured point clouds are not changing the DSP parameters to improve the downstream vision models.

In this work, we propose an optimization method for LiDAR DSPs that minimizes end-to-end domain-specific losses such as RMSE depth against ground truth depth, and downstream IoU metric measured after 3D object detection. To assess the proposed method we devised a LiDAR simulation method based on the CARLA engine [1] that models the full transient noisy waveform formed by multiple laser echoes and a subsequent DSP. Next, we optimize DSP hyperparameters by solving a black box Multi-Objective Optimization (MOO) problem with a novel active CMA-ES [27] that relies on the scale-invariant max-rank multi-objective scalarization loss [28] to guide the solver toward balanced solutions. This novel CMA-ES variant changes centroid weights every generation, alternating between those of [28], empirically found to "eagerly" follow gradients, and those of [29], designed to stabilize boundary minima and empirically found to stabilize the search near minima. The novel CMA-ES also adds the computation of the loss of every generation's centroid to that of the usual randomly drawn ones using them to greedily speed up the computation when they are a "best so far" without corrupting the CMA-ES statistics that guide convergence. The proposed method finds a set of best compromise solutions over multiple objectives within the Pareto front, focusing on solutions for which no loss component has a comparatively poor value. We validate the proposed optimization method for 3D object detection and depth estimation. For these applications, we aim to improve end-to-end losses compared to manual tuning and standard approximation-based approaches. Specifically, we make the following contributions:

- We introduce a full-wave LiDAR simulation model for the CARLA simulation environment that outputs full waveform data where a realistic BRDF is used to model intensity reflections.
- We devise an optimization method for LiDAR DSP algorithms and their respective hyperparameters with a novel CMA-ES optimization algorithm.
- We validate end-to-end DSP optimization for 3D object detection and depth-estimation. For all applications, our approach outperforms existing state-of-the-art methods, including manual expert tuning.

Since commercial LiDAR units are black boxes and protected by intellectual property rights, it is not straightforward to adapt existing virtual engines like CARLA to generate transient waveforms and model LiDAR DSPs. For that reason, in some situations, noise models, DSP behaviour and light physical interactions might not be completely accurate compared to real LiDARs. Furthermore, the DSP parameters designed for this work might not reflect their real counterparts as their effects and numbers might be quite different even between real LiDAR units.

3.2 Related Work

A growing body of work has explored optimizing sensors for downstream vision tasks. Existing methods address camera ISPs and optics [28]–[31]. Specifically, departing from manual tuning of hyperparameters by experts in the sensor design phase, these methods explore approaches that optimize these parameters automatically, driven by a downstream loss function. As sensor hyperparameters can be categorical and the loss is often non-convex and even noisy, a diverse set of optimization methods have been proposed to tackle this approach. Early methods proposed to optimize specific processing blocks [32]–[34] as differentiable programs or in a reduced parameter space. Recent methods employ differentiable proxy functions [30] or use 0th order optimization with alternating optimization algorithms for downstream detectors [28], [29]. While all of these existing methods are successful for camera pipeline optimization, from the optics [31] to downstream detectors [29], to our knowledge, no existing methods have been proposed for LiDAR systems. As such, existing LiDAR sensors are designed in isolation from the downstream detection or depth estimation problem – this work aims to close this gap and proposes an approach for domain-specific LiDAR hyperparameter optimization.

LiDAR sensors generate point clouds by emitting pulses of light into a scene and they measure the round-trip time-of-flight of the returned light arriving at the sensor. Extracting a point cloud from the time-of-flight measurements is a complex process and depends on the specific measurement procedure, including beam formation, scanning, pulse/continuous wave generation and the final peak finding within the acquired temporal laser response [35], [36]. LiDAR sensors differ in their scanning pattern, beam steering technology, wavelength, pulse profile, coherence of the measurement step, detector technology and DSP capabilities processing the measurement echoes [35], [37]. Existing LiDAR systems can extract single or multiple peaks resulting from multi-path interreflections in the scene. Assuming a single Lambertian reflector in the scene, the temporal resolution and SNR of the measurement is fundamentally coupled to laser power. To this end, existing optimization methods [38] explored automatically adjusting laser power during runtime in order to maximize SNR while preventing oversaturation. Other approaches have been proposed for adaptative beam steering [39], [40]. Recently, Vödisch et al. [41] proposed beam configuration optimization via reinforcement learning methods driven by a downstream 3D detection loss.

This method is the most similar approach to the proposed method but solves a different problem. While Vödisch et al. aim to predict the beam patterns, i.e., where to place the sparse LiDAR samples, we optimize for optimal LiDAR DSP parameters independently of the scanning patterns.

To test and validate the proposed method, we introduce a new LiDAR simulation method that plugs directly into the CARLA simulator. A range of existing simulation environments have been proposed [42]–[44], [45] with the open-source CARLA [1] simulator particularly being broadly adopted in the field. These simulation frameworks have allowed multimodal synthetic datasets including PreSIL [26], SHIFT [24], AIODrive [25], [46] and SYNTHIA [47]. Although these datasets have been employed successfully by researchers and industry, the underlying simulation methods employ heuristic LiDAR forward models and none of the datasets include full waveform returns that would allow us to simulate the LiDAR point generation. Even the AIODrive dataset [25] where multiple peaks are returned via a depth image convolution and Single-Photon Avalanche Diode (SPAD) quantization already bakes the transients into the SPAD image formation, and hence does not allow us to simulate realistic transients for LiDAR DSP optimization. To address this simulation, we propose a simulation approach that uses a full waveform generated with a realistic BRDF processed with a virtual DSP to give a final point cloud.

3.3 LiDAR Forward Model

Consider a single laser pulse emitted by a LiDAR unit onto a 3D scene from which a returned signal is detected by a SPAD detector that then sends temporal histograms to the sensor DSP. We model the temporal photon flux $\psi^{(n)}$ [48]–[50] incident on the sensor at time t for channel n as

$$\psi^{(n)}(t) = \left(H * g^{(n)}\right)(t) + a(t),$$

where $g^{(n)}$ is the initial temporally varying photon flux emitted by the laser, H the transient response from the scene, $a(t)$ the ambient photon flux and * is the temporal convolution operator. Note that the transient scene response H, by definition, includes multi-path returns from interreflections and scattering in the scene. The detector measures the photons and digitizes them into temporal wavefronts that are processed by the DSP. For low photon counts or path lengths beyond a few meters like in typical automotive scenes, the binning process can be modeled as a random process following a Poisson distribution [38], [48]–[50]. We thus model the number of detected photons $r^{(n)}$ for channel n in a given time bin k of the wavefront as

$$r^{(n)}[k] \sim \text{Poisson}\left(\int_{k\Delta}^{(k+1)\Delta} \psi^{(n)}(t)dt\right),$$

where Δ is the integration time per bin.

3.3.1 Transient Scene and Pulse Model

In the context of LiDARs, Rasshofer et al. [36] introduced a linear model for direct laser reflections that models the incident transient response $H * g^{(n)}$. Adopting this model allows to write

$$\left(H * g^{(n)}\right)(R) = C \int_0^{2\tau^{(n)}} g^{(n)}(t) H\left(R - \frac{ct}{2}\right) dt,$$

where R is the distance between the sensor and the observed point, c is the speed of light, C a proportionality constant independent of t and R describing the system and $2\tau^{(n)}$ is the total pulse duration. The conversion $t = R/c$ translates the path length to time. Like Hahner et al. [20] and Rasshofer et al. [36], we adopt the following pulse shape

$$g^{(n)}(t) = \begin{cases} P_0^{(n)} sin^2\left(\frac{\pi t}{2\tau^{(n)}}\right) & 0 \le t \le 2\tau^{(n)} \\ 0 & \text{otherwise} \end{cases},$$

where $P_0^{(n)}$ is the pulse power magnitude. Meanwhile, the transient scene response H consists of the geometric attenuation of the light which is proportional to $1/(2R)^2$ and the response from the objects scattering the light. For a single opaque object labelled i, the latter is proportional to its reflectance ρ_i and to a Dirac function $\delta(R - R_i)$ where R_i is its distance to the LiDAR. Reformulating the equations above, for a single echo from i yields

$$\left(H * g^{(n)}\right)_i(R) = \begin{cases} f_i^{(n)}(R) & R_i \le R \le R_i + c\tau^{(n)} \\ 0 & \text{otherwise} \end{cases},$$

with

$$f_i^{(n)}(R) = \frac{CP_0^{(n)}\rho_i}{4R_i^2} sin^2\left(\frac{\pi}{c\tau^{(n)}}(R - R_i)\right).$$

3.3.2 Object Reflectance Model

An object's reflectance ρ_i depends on many factors such as the material BRDF and the angle of incidence θ with respect to the surface normal. We approximate this term based on the integrated specular and diffuse components of the retro-reflected portion of the Cook-Terrance reflectance model [51]:

$$\rho_i = \frac{\alpha^4 scos\theta}{4[cos^2\theta(\alpha^4 - 1) + 1]^2[cos\theta(1 - k) + k]^2} + dcos\theta.$$

$s, d, \alpha \in [0,1]$ define respectively the specular, diffuse and roughness properties of a material while $k = (\alpha + 1)^2/8$. In lieu of a large texture database, we abuse the Phong-like parameters s, d, α used in CARLA [1] to realistically render the textures. While these parameters are normally not directly accessible, we extract them by projecting the targeted hit points on custom camera images encoding these values. An example of such images is shown in Figure 3.2. To be more precise, if we define these type of image as $I_{sd\alpha}$ and defining a function P_i that takes an image as input and returns the pixel information at the projected object i's location, then we write $[s, d, \alpha] = P_i(I_{sd\alpha})$.

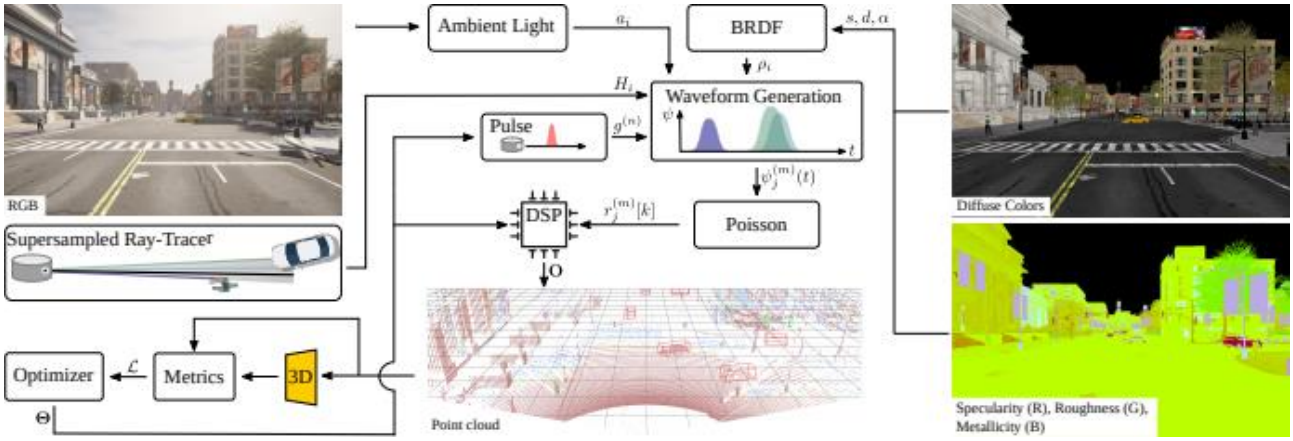

Figure 3.2: LiDAR Simulation Method. We propose a parameterizable LiDAR simulation model that generates full transient waveforms by extracting scene response H, ambient light a and object reflectances s, d, α from CARLA. The optimized vector Θ includes both pulse and DSP hyperparameters. Wavefronts are processed by the DSP, resulting in a point cloud O. We "close the loop" by feeding O to 3D object detection and depth estimation methods that define loss functions L when evaluated on validation datasets. These end-to-end loss functions drive an MOO solver which converges to optimal parameters.

3.3.3 Ambient Illumination

The next step is to model the ambient light illumination $a(t)$. We do this by taking the same approach as [25] and project the targeted object i's location on the red channel of a fully rendered RGB camera image where shadows and reflections are all taken into account. We label this kind of image as I_{red} and, for simplicity, we approximate $a(t)$ to be constant over the whole waveform time bins. Thus, we can write $a(t) \equiv a_i = P_i(I_{\text{red}})$ where a_i is constant $\forall t$.

3.3.4 Multi-path Transients

We simulate multi-path transients for laser beams hitting object discontinuities as primary effect for automotive LiDAR scenarios. These effects are modeled as linear combination of neighboring waveforms. To this end, a super-sampled collection of $\{R_i\}$ is computed using direct illumination only. Then, a downsampled waveform ψ_j for each LiDAR channel and each horizontal angle is generated with

$$\psi_j^{(m)}(R) = \sum_{i \in N(j), n \in N(m)} K_i^{(n)} \left(H * g^{(n)}\right)_i (R) + a_i,$$

where $N(j)$ and $N(m)$ respectively define the spatial neighborhood of target point j and target channel m and $K_i^{(n)}$ are normalized weights that can be interpreted as a beam spatial intensity profile.

3.3.5 DSP Model

Once a waveform $\psi_j^{(n)}(R)$ is generated for a given j and m, it is then sampled and passed to the DSP unit. The latter can generally be viewed as a functional $\Phi_j^{(m)}\left(\Theta, r_j^m\right)$ of some hyperparameter set Θ that takes this final sampled waveform $r_j^{(m)}$ as input and returns a 3D point alongside a reflectivity value. The DSP uses multiple processing steps which includes denoising, peak finding and thresholding among others and

these processes might employ both continuous and discrete variables within $\{\Theta\}$. There exist multiple algorithms to generate a point cloud out of a waveform signal [52]. We focus on one of the simplest: the rising edge method where a point is registered every time the amplitude of the waveform goes over a certain threshold $T \in \{\Theta\}$ thus generating a list of potential return candidates if multiple peaks are present in the waveform. The returned point can then be picked based on a discrete return mode parameter $M \in \{\Theta\}$. Figure 3.3 shows an example DSP pipeline where each processing step is shown as a separate block.

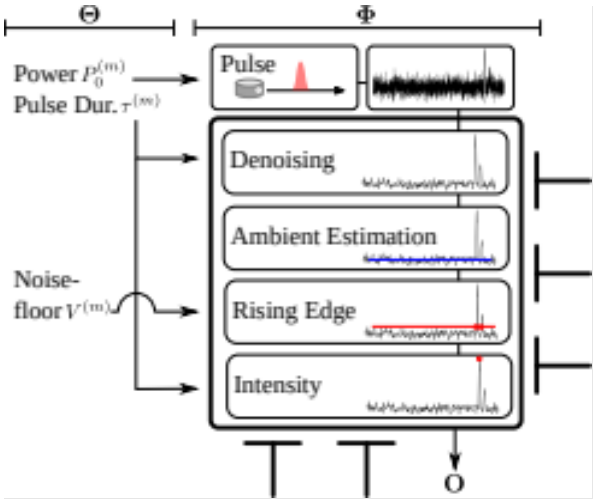

Figure 3.3: LiDAR Sensing and DSP Model Φ. The LiDAR sensing and DSP hyperparameters Θ, including the pulse power, duration, and parameters for the rising edge detection in a transient wave-front, affect the quality of the generated point cloud O.

3.4 Optimization

The proposed MOO method yields high quality point clouds with optimal mean Average Precision (mAP) when fed to the downstream object detection and classification module.

3.4.1 LiDAR Optimization

Like Mosleh [28], we tackle hyperparameter value selection through loss-driven optimization. Like Robidoux [29], we optimize the system as a whole, bundling hardware functionality—each channel's laser beam power—with the DSP's. The operation of the LiDAR imaging pipeline Φ is modulated by P hyperparameters $\Theta = (\Theta_1, \dots, \Theta_P)$ with continuous range of values normalized by relaxation and remapping to the unit interval $[0,1]$ [28], [29]. Hardware registers actually have discrete ranges of values, for example $\{0, \dots, 2^{14} - 1\}$ for an effectively continuous hyperparameter and $\{0,1\}$ for an algorithmic branch toggle [30]. With $T \gg 2\tau$, we model each of the J channels Φ_j of $\Phi = (\Phi_1, \dots, \Phi_J)$ as

$$\Phi_j : [-T, 0]^{[0,\infty)} \times [0,1]^P \to S^{[0,\infty]}, (r_j, \Theta) \mapsto O_j,$$

O_j being a mapping from the unit sphere S (a proxy for 3D projective geometry) to nonnegative distances with ∞ understood as "undefined" so that each Φ_j reconstructs a piece O_j of the overall point cloud O from a waveform r_j truncated to the time interval $[-T, 0]$. The overall Θ-modulated pipeline Θ maps the set of truncated wave forms $r = (r_1, \dots, r_J)$ to the point cloud O: Φ: $(r, \Theta) \to O.O$ contains the compressed

information available to downstream detectors. We seek solutions $\Theta \quad = argmin_{\theta\in[0,1]^P} L(\Theta)$ which are Pareto-optimal with respect to the MOO loss vector $L = (L_1, \ldots, L_L)$. Loss components may not be end-to-end: L_l could be computed from a result tapped out of the pipeline (e.g., a channel's waveform r_j); from the point cloud O (e.g., SNR); or from the output of a downstream detector O (e.g., a deep Convolutional Neural Network (CNN)) which ingests O (e.g., mAP).

The set of solutions is the *Pareto front* [53], which contains Pareto-optimal compromises between the various losses, that is, hyperparameter vectors which are not Pareto-dominated by another.

3.4.1.1 Monotone Max-Rank Loss Scalarization

Scalarizations are used to combine multiple objectives so that a single objective optimizer yield solutions to MOO problems [54], the best known being the *convex combination* $\sum_{l=1}^{L} w_l\, L_l$. With such scalarizations, the weights w_l are difficult to choose when losses' variations are not commensurate. This is less of an issue with the *max-rank loss*, a Chebyshev scalarization based on ranks introduced by Mosleh in [28]. The max-rank loss is dynamic: For a fixed hyperparameter its value increases as additional data is acquired. Nonetheless max-rank loss minimizers are automatically Pareto-optimal (relative to available data). Within the Algorithm described below, the weighted max-rank loss $M_q^{(m)}$ of the hyperparameter vector $\Theta_q^{(m)}$ at the end of generation n is $M_q^{(m)} = \max_{l\in\{1,\ldots,L\}} \left\{ w_l \cdot \text{rank of } L_q^{(m)} \text{ within } \{L_r^{(o)}\}_{o\in\{1,\ldots,n\}, r\in\{0,\ldots,4P\}} \right\}$, with ranks starting at 0 and loss component value ties resolved by left bisection. Because the weights w_l multiply ranks, they are non-dimensional thus better suited to dialing in the relative importance of various loss components. The max-rank loss is dynamic, with values, for a given hyperparameter vector, monotone non-decreasing with respect to the addition of data. Because the weights multiply ranks, they are non-dimensional, thus suited to dialing in the relative importance of various loss components. Mosleh et al. Propose that user-defined weights for each loss component be multiplied by the proportion of individuals that ``fail'' to pass a user-defined threshold. We find that these weights, which break monotonicity, do not lead to better convergence. We use fixed, unit by default, loss weights.

3.4.2 Dual-Weighted CMA-ES

Our algorithm differs from the max-rank loss-based CMA-ES optimizers of [28], [29] as follows. It alternates between the gradient-seeking weights of [28], which assign zero weight to the worst quarter of each generation instead of the usual half so as to exploit the symmetry of Gaussian probability distributions in the second and third quartiles to get a more accurate gradient approximation; and the boundary stabilizing weights of [29], with no discard so that further exploration not go in the wrong direction near non-smooth local minima. With active CMA-ES methods (with centroid weights monotone as functions of the rank and with negative values [55]) that discard some proportion of each generation's individuals, as is standard, the solver is easily tricked into exploring in the wrong direction near local minima because the very worst direction implicitly gets 0 centroid weights while less "bad" directions get strictly negative weights. For this reason, the "stabilizing" weights do not discard. The alternating weights are used with larger generations, namely $4P$, than in [28] ($4P/3$) and [29] ($2P$). Per [56], larger generations are preferable with rough or noisy losses, as we verified with A/B testing.

Another novel feature of the proposed CMA-ES variant is that the loss of the weighted centroid of every generation is computed, which provides higher quality transients as the solver explores the Pareto front. So

as not to corrupt the standard CMA-ES statistics (based on one random Gaussian point cloud per generation), the centroid's loss does not affect their computation. However, if it or any other individual of the generation happens to be a max-rank loss minimizer, it becomes the next generation's Gaussian cloud center (ties within the generation are resolved by selecting the minimizer closest to the ties' centroid).

Algorithm 1 Lidar Hyperparameter Optimization.

Require: Lidar $\mathbf{\Phi}$, $\mathbf{\Theta} \in [0,1]^{\mathrm{P}}$ (initial hyperparameter vector),
$\mathrm{N} \in \mathbb{N}^*$ (number of generations), $\varepsilon \in (0, 1/3)$ (small bound),
$\mathbf{C} \in \mathbb{R}^{\mathrm{P}\times\mathrm{P}}$ (CMA-ES "directional" covariance matrix factor),
$\sigma \in [\varepsilon, 1/3]$ (square root of covariance matrix "scale" factor)

$\Lambda \leftarrow \sqrt{\mathrm{P}}/3$ (large bound: $\sqrt{\mathrm{P}}$ = diameter of the unit hypercube)
$\mathbf{p} \leftarrow \mathbf{0}$, $\mathbf{c} \leftarrow \mathbf{0}$ (CMA-ES path vectors), $\mathbf{\Theta}_{\text{center}} \leftarrow \mathbf{\Theta}$
for $n = 1$ **to** N **do**
  $\mathbf{\Theta}^{0,n} \leftarrow \mathbf{\Theta}$, $\mathbf{C} \leftarrow (\mathbf{C} + \mathbf{C}^\intercal)/2$, $\sigma \leftarrow \mathbf{median}(\varepsilon, \sigma, 1/3)$
  if smallest $\mathbf{C}$ eigenvalue $\lambda_{\min} > 1$ **then**
    $\mathbf{C} \leftarrow \mathbf{C}/\lambda_{\min}$, $\mathbf{c} \leftarrow \mathbf{c}/\sqrt{\lambda_{\min}}$, $\sigma \leftarrow \mathbf{min}(\sqrt{\lambda_{\min}}\sigma, 1/3)$
  end if
  if smallest $\sigma^2\mathbf{C}$ eigenvalue $\lambda_{\min} < \varepsilon^2$ **then**
    $\sigma \leftarrow \mathbf{min}(4\sigma/3, 1/3)$
  end if
  if smallest $\sigma^2\mathbf{C}$ eigenvalue $\lambda_{\min} < \varepsilon^2$ **then**
    Push $\mathbf{C}$ toward the identity by clamping its eigenvalues to $\left[\varepsilon^2/\sigma^2, \infty\right)$ and replacing it by its square root
  end if
  if largest $\mathbf{C}$ eigenvalue $\lambda_{\max} < 1$ **then**
    $\mathbf{C} \leftarrow \mathbf{C}/\lambda_{\max}$, $\mathbf{c} \leftarrow \mathbf{c}/\sqrt{\lambda_{\max}}$, $\sigma \leftarrow \mathbf{max}(\sqrt{\lambda_{\max}}\sigma, \varepsilon)$
  end if
  if largest $\sigma^2\mathbf{C}$ eigenvalue $\lambda_{\max} > \Lambda^2$ **then**
    Push $\mathbf{C}$ toward the identity by replacing it by its square root and clamping its eigenvalues to $\left(-\infty, \Lambda^2/\sigma^2\right]$
  end if
  $\boldsymbol{\mathcal{L}}^{0,n} \leftarrow$ losses for Lidar $\mathbf{\Phi}$ modulated by $\mathbf{\Theta}^{0,n}$
  for $p = 1$ **to** $4\mathrm{P}$ **do**
    $\mathbf{\Theta}^{p,n} \leftarrow$ random draw from Gaussian distribution with covariance matrix $\sigma^2\mathbf{C}$ centered at $\mathbf{\Theta}_{\text{center}}$
    $\mathbf{\Theta}^{p,n} \leftarrow \mathbf{\Theta}^{p,n}+$ Gaussian distribution with diagonal covariance matrix proportional to the square of individual hyperparameters' quantization grain
    $\mathbf{\Theta}^{p,n} \leftarrow \mathbf{\Theta}^{p,n}$ reflected back into $[0,1]^{\mathrm{P}}$
    $\boldsymbol{\mathcal{L}}^{p,n} \leftarrow$ losses for LiDAR $\mathbf{\Phi}$ modulated by $\mathbf{\Theta}^{p,n}$
  end for
  Compute $\{\mathcal{M}^{q,m,n}\}_{q\in\{0,\ldots,4\mathrm{P}\},m\in\{1,\ldots,n\}}$ by including $\{\boldsymbol{\mathcal{L}}^{p,n}\}_{p\in\{0,\ldots,4\mathrm{P}\}}$ in rank computations
  Use "eager" centroid weights with $\lambda = 4\mathrm{P}$, $\mu = 3\mathrm{P}$
  if n is odd **then**
    Use "stable" centroid weights with $\lambda = \mu = 4\mathrm{P}$
  end if
  Standard CMA-ES update of $\mathbf{\Theta}$, σ, $\mathbf{C}$, $\mathbf{p}$, $\mathbf{c}$ based on $\{\mathbf{\Theta}^{p,n}\}_{p\in\{1,\ldots,4\mathrm{P}\}}$ and $\{\mathcal{M}^{p,n,n}\}_{p\in\{1,\ldots,4\mathrm{P}\}}$ except that if the $\mathcal{M}^{p,n,n}, p \in \{1,\ldots,4\mathrm{P}\}$ are all equal also multiply $\mathbf{C}$ by $4/3$ and σ by $\sqrt{4/3}$
  $\mathbf{\Theta}_{\text{center}} \leftarrow \mathbf{\Theta}$
  if $\min_{p\in\{0,\ldots,4\mathrm{P}\}} \mathcal{M}^{p,n,n} < \min_{q\in\{0,\ldots,4\mathrm{P}\},m\in\{1,\ldots,n\}} \mathcal{M}^{q,m,n}$ **then**
    $\mathbf{\Theta}_{\text{center}} \leftarrow$ minimizer closest to centroid of minimizers (distance ties resolved arbitrarily)
  end if
end for
return $\mathcal{M}^{p,n,\mathrm{N}}$ (weighted max-rank loss) minimizer (ties included) within the the Pareto front (subset of $\{\mathbf{\Theta}^{p,n}\}_{p\in\{0,\ldots,4\mathrm{P}\},n\in\{1,\ldots,\mathrm{N}\}}$ that contains hyperparameter vectors that are not Pareto-dominated)

Figure 3.4: Black box LiDAR optimization algorithm, a variant of single objective Covariance Matrix Adaptation-Evolution Strategy that uses the max-rank loss MOO scalarization.

3.5 Results

As a start, we explore channel-wise laser power and rising edge threshold optimization using our optimization algorithm. We assign independent power level $P_0^{(j)}$ and rising edge threshold $V^{(j)}$ (see Section 3.3.5) for each LiDAR channel j.

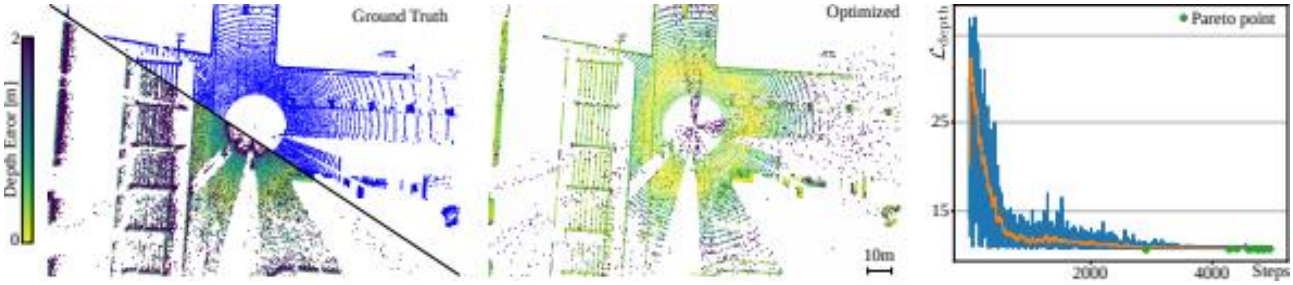

Figure 3.5: The color encodes the individual depth error of each point clipped at 2m. Convergence graph shows average RMSE loss Ldepth from Eq. (14) against optimization step with Pareto points locations for the proposed method.

Combined with denoising parameters, all of these values form the ensemble Θ used to define the DSP in this experiment. We optimize the point cloud depth RMSE averaged over multiple frames using the following loss function:

$$L_{depth}(\Theta) = \frac{1}{F}\sum_{f=1}^{F} RMSE\left(R_f, \Phi_f^{(R)}\right)$$

where F is the number of frames in the validation set,$M_f fr_i^f if, R_i^f$ Note here that missing points are treated like they have a depth of 0. We defined the power levels as discrete parameters spanning only 7 values where the lowest makes the peak almost indistinguishable from ambient noise while the highest have a high chance of saturation at close distances. Necessarily, cranking up the power levels for each channel would increase SNR and decrease the depth error (L_{depth}). However, in order to avoid as much as possible saturated intensity readings, we added the intensity loss function

$$L_{intensity}(\Theta) = \frac{1}{F}\sum_{f=1}^{F} RMSE\left(I_f, \Phi_f^{(I)}\right).$$

Optimized point cloud results after >5000 steps are shown in the middle of Figure 3.5 compared to the initial configuration on the left. Ground truth is overlayed on the middle upper half showing the optimization target depths. The graph on the right shows the depth loss $L_{depth}(\Theta)$ vs the optimization steps. Pareto points are shown as orange squares and the initial configuration loss is depicted by the green circle. The orange line is a running average over 500 steps and acts as a guide to the eye showing the overall decrease over time of the loss function. Table 3.1 shows the loss values for final Pareto point for this run compared to other state-of-the-art MOO optimization algorithms. The findings validate that our proposed method outperforms other optimization algorithms. Our method seeks one single balanced configuration, which we take to be the last Pareto point of a run. Comparing with the final Pareto points found by other algorithms, we see that they are not as balanced.

Table 3.1: LiDAR Depth and Intensity MOO. In the table, final Pareto points are ordered. The proposed method finds a balanced Pareto point which ranks 5 out of 10 with respect to both $RMSE_{depth}$ and $RMSE_{intensity}$, balance also evident by it minimizing the ℓ1-norm of the losses (bold).

Method	$\boldsymbol{RMSE_{depth}}$	$\boldsymbol{RMSE_{intensity}}$	$\boldsymbol{\|Loss\|_1}$
Expert-Tuned	12.15	1.95	14.10
C-TAEA	9.20	2.37	11.57
NSGA-III	9.46	2.14	11.60
U-NSGA-III	9.82	2.03	11.65
R-NSGA-III	9.78	3.18	12.96
Proposed	9.86	1.56	**11.42**
SMS-EMOA	10.16	1.43	11.59
AGE-MOEA	10.17	1.29	11.46
Mosleh et al.	10.98	1.01	11.99
RVEA	30.30	0.67	30.97

4 Gating Parameter Optimization

The output of the gated camera is determined by a set of user-adjustable gating parameters. However, the details of its inner-workings are not revealed to the user. Therefore, the gated imaging system can be considered as a black box unit.

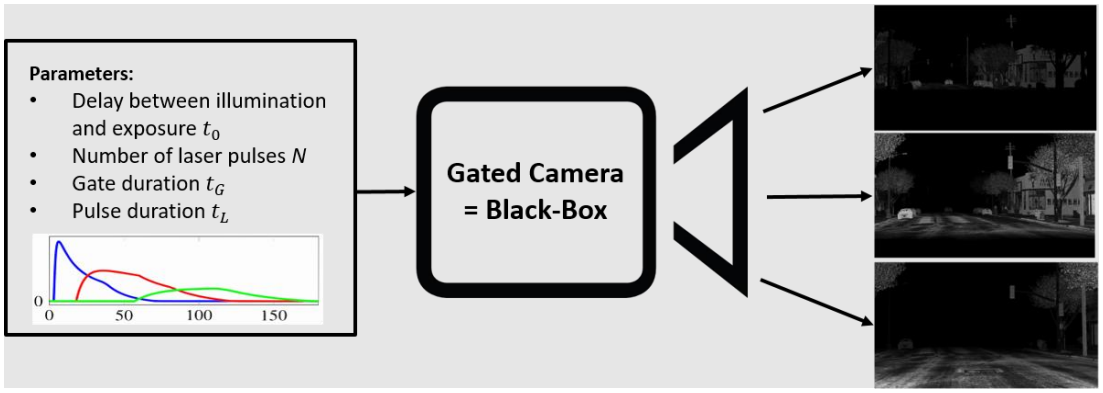

Figure 4.1: Configurable parameters of a black box Gated camera.

As shown in Figure 4.1, the output image of the gated camera can be influenced by adjusting the delay between illumination and exposure t_I, the number of laser pulses N, the gate duration t_G and the pulse duration t_L. These parameters determine the range-intensity-profiles which represent the distance-dependent intensity values of the output image and therefore which distance of the scene is illuminated. By varying the gating parameters, it is possible to capture multiple images of the same scene where different areas of the scene are illuminated.

For depth estimation, currently, three images covering different ranges are captured for each scene [57]. The parameters are chosen in such a way that a range of 0-150m is covered and the individual images, also called slices, partially overlap in the range they capture. However, the parameters were selected manually and are not customized for any specific task or environmental condition. As a result, the capabilities of the gated camera are not fully exploited, and it is expected that current methods for depth estimation can be improved by optimizing the gating parameters for this use case.

This work addresses the extent to which a gated camera's parameters can be optimized for depth estimation. In particular, the performance improvement of gated depth estimation with synthetic data created to simulate in clear conditions at night shall be achieved by optimizing the parameters for this condition. To determine the improvement, depth maps are created for synthetically created scenes with unoptimized and optimized parameters and compared with popular metrics used for the evaluation of depth estimation approaches. For the optimization, a framework is created that uses deep learning methods, can be used with synthetic and real-world data, and is extensible to optimize other conditions and tasks.

The method follows the approach of [30] that uses a neural network as a differentiable proxy function to perform the optimization in two stages. The decisive advantage is that only a limited amount of data has to be collected in advance, while optimal setting parameters can still be found with a very high degree of reliability.

4.1 Dataset

For the parameter optimization a synthetic dataset is used due to several reasons. First, a dataset with real gated images that meets the requirements does not exist. Collecting this data is very cumbersome and is done after the initial proof of concept. Second, a suitable framework for creating synthetic data has already been presented by [58]. Also, a synthetic dataset is much more flexible, so changes can easily be made to the dataset during experiments. Furthermore, ground truth annotations regarding, for example, the depth are included and are guaranteed to have no deviation.

The gated simulation framework provides the ability to extract and synthesize gated image data from the open-world-action-adventure video game Grand Theft Auto-V (GTAV). For the synthesis of the gated images, gated parameters and daytime can be chosen as desired which gives the possibility to create different gated slices from the same scene.

To create the gated images, diffuse reflectances, normal vectors, specular reflectance, glossiness, sunlight illumination, active lights, depth maps, and the projection matrix are extracted while playing. First, the GTAV camera view is adapted to the intrinsic parameters of the gated camera prototype. The laser and gated camera prototype specifications are chosen to represent the BrightWay Vision BrightEye™ [59], a gated camera system developed for the automotive market. The adaption is done by projecting each pixel into 3D space with the projection matrix and then back into the gated frame.

Since the gated camera operates in the Near InfraRed (NIR) spectrum and GTAV does not provide any NIR reflectance or NIR image formation model, the NIR reflectance is approximated based on an empirical NIR model. This is done by inverting the original RGB image and overlaying the original and the inverted image by selecting the maximum for each pixel in each channel. Then the image is transformed to be monochrome. Finally, a gamma correction is applied to darken the image. Active illumination is simulated as laser illumination being emitted from an illumination source under the vehicle's front bumper. The resulting shadows are simulated by transforming 3D points into a laser coordinate system and then determining which points are illuminated and which are not. Gating is applied according to the following gating equation:

$$I(r) = \alpha C(r) = \int_{-\infty}^{\infty} g(t - t_0)\kappa(t; \alpha, r)dt$$

Here, $I(r)$is the exposure measure of an object with reflectivity (albedo) α. $C(r)$ is the range intensity profile. $\kappa(t; \alpha, r)$ describes a delayed and attenuated version of the laser pulse emitted by the active light source and $g(t - t_0)$ is a rectangular function which represents the gain of the detector.

The gating parameters can be selected freely. Open parameters such as photon-to-pixel conversion gain, delays due to signal run-time, and the dark level of the camera were calibrated with measurements at night on targets with defined reflectivity. Passive daylight components are added according to the exposure duration and ambient illumination. Additionally, saturation and blooming effects are simulated, and a noise model is applied. Finally, the image is clipped to the sensor bit depth. The schematic gated image formation can be seen in Figure 4.2.

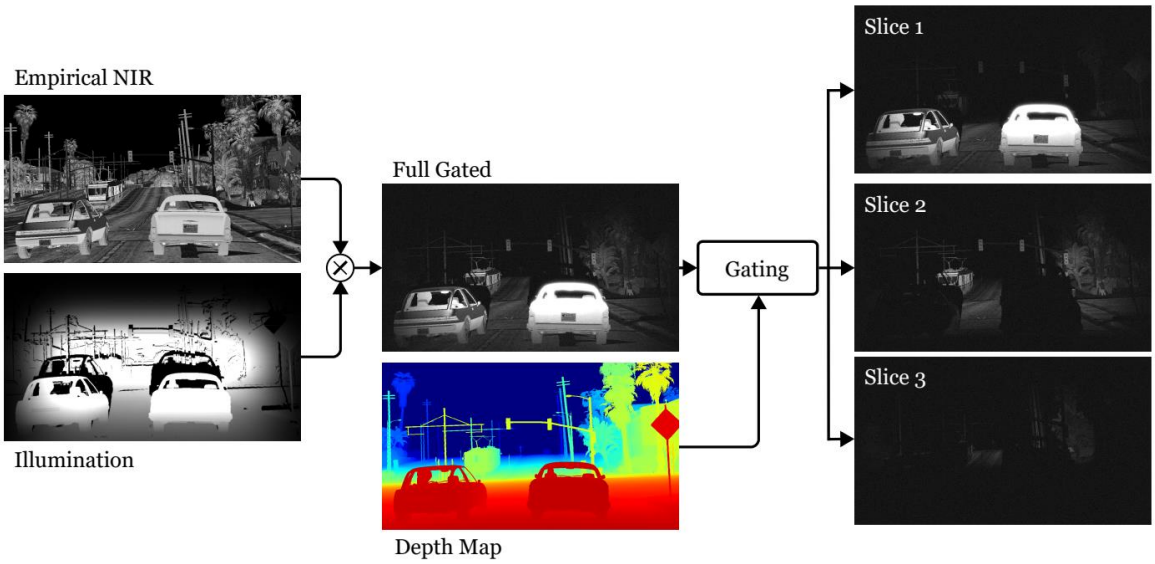

Figure 4.2: Gated image formation. From simulated illumination and the empirical NIR image, a full gated image can be constructed. The three slices are obtained by applying gating with the predefined parameters, the corresponding depth map and t to obtain the final image, saturation, blooming and noise is added.

9778 of the sample scenes extracted by Gruber et al. [57] with an image resolution of 1280 × 720 pixels and represented using 10 bits are adopted. 7655 samples are used for training, 500 for evaluation and 1623 samples for testing. All samples are simulated at night. The gated parameters used to create the different slices are generated with Latin-Hypercube sampling within a predefined range for each parameter. 40 parameter settings are used for training and 20 for testing.

In addition, LiDAR data is required for the optimization. The gated simulation framework does not provide tools for LiDAR simulation, so this was added. The LiDAR is located at another location than the camera, resulting in shadows where no LiDAR-points are given. First, from the gated simulation framework extracted 3D points are transformed into a LiDAR coordinate system which is chosen to be located at the top of the car. Based on this coordinate system, a scanning LiDAR of 64 rows with an angular resolution of 0.4° and a vertical field of view of 25° is modelled. A direction vector is calculated for each emitted light beam. This is matched with the normal vectors, also extracted from GTAV. A LiDAR point is set, by assigning the corresponding depth value which is given in the depth map, if the vectors match. Only the point closest to the LiDAR source is set if there are multiple matches. Then, the resulting LiDAR depth map is transformed back into the camera coordinate system. The corresponding depth-value is assigned at every sampled LiDAR-point. Resulting LiDAR data and the corresponding depth maps can be seen in Figure 4.3.

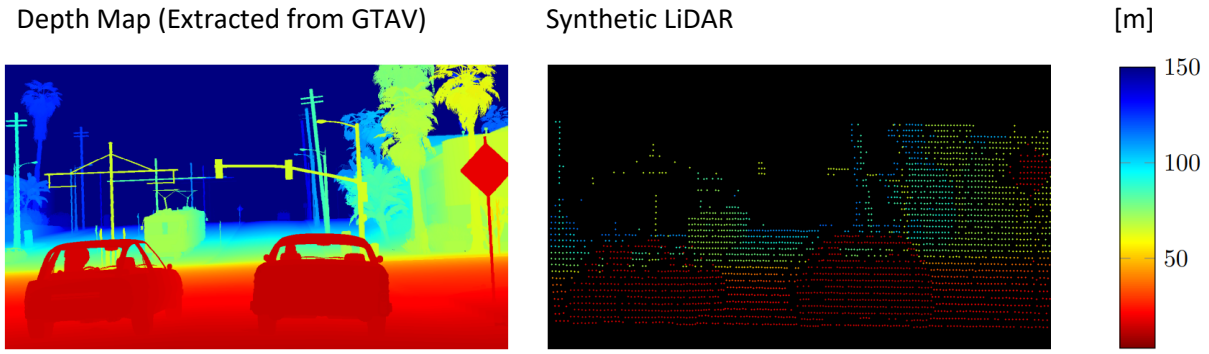

Figure 4.3: Synthetic LiDAR and depth map. The points of the LiDAR are enhanced for better visibility.

4.2 Method

Even though synthetic data is used in this work, the method is developed for the case of real gated data that is captured by a real gated camera. This has the advantage the method can be adapted to real-world applications without much adaption effort. The method's main idea is to consider the gated imaging system as a black box and optimize its parameters, also called hyperparameters, using a model-based, also called surrogate-based, approach which bypasses the imaging system, see Figure 4.4. Optimization with a model-based method is chosen because modelling the gradients allows for a much more precise optimization than gradient-free methods. For the realization, the approach of Tseng et al. for hyperparameter optimization in black box image processing is adapted [30]. Therefore, a two-stage neural network is implemented. A proxy function is trained in the first stage to learn the mapping between gating parameters and camera output. This enables the modification of full gated images to produce realistic gated images based on provided gated parameters. The proxy outputs are simulations of images that would be created if the gated camera would capture images with the given parameters. In a second step, the weights of the proxy network get fixed, and it is used to generate training data for the second stage. The network of the second stage is trained to fulfil the task of generating a depth map of the respective scene. Since the proxy function of the black box is differentiable, the parameters can be optimized to create the best possible depth map using a gradient-based optimizer.

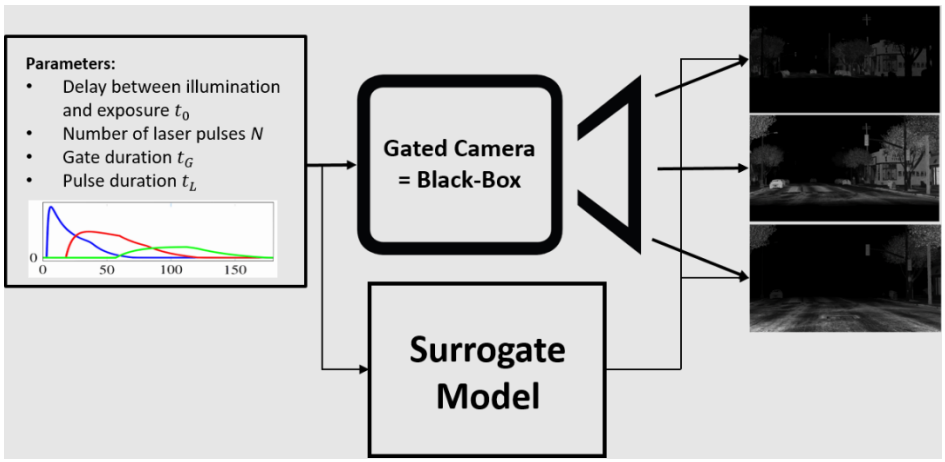

Figure 4.4: Black box with surrogate model which is used to bypass the black box during the optimization.

4.3 Concept

Following the notation of Tseng et al. [30], the gated camera can be represented by the function f_{gated} that maps the observed scene S to the image O_{gated} where only a predefined range is visible. The visible range is dependent on the parameters P. Therefore, the output image is defined by $O_{gated} = f_{gated}(S, P)$. The hyperparameter optimization problem can be mathematically expressed by

$$P \quad = arg \min_{\{P_1,P_2,P_3\}} \sum_{i=1}^{N} L_{depth}\left(f_{depth}\left(f_{gated}(S_i, P_1), f_{gated}(S_i, P_2), f_{gated}(S_i, P_3)\right), T_i\right)$$

where the optimal parameters $P \quad = \{P_1 \ , P_2 \ , P_3 \ \}$ are sought by optimizing a depth-specific performance metric. This can be modelled as the minimization of the loss function L_{depth} that measures the deviation between the function fdepth, which maps gated images to a corresponding depth map, and known ground truth target depth maps T_i for N scenes. Neural networks are the most successful approach for recovering depth from gated images. Therefore, the method from Gruber et al. is adapted [57] which derives the depth from three gated images. To perform this mapping, a function f_{depth} is created. It uses three gated slices with overlapping ranges that are captured with a fixed set of gated parameters $P_i \in P$ for $i = 1,2,3$ to compute the corresponding depth map.

Step 1: Training the Differentiable Proxies Function to learn the Gated Function

Since the gated camera is a black box system, the only way to interact with it is to evaluate f_{gated} on example scenes with specific parameter settings. In order to solve the optimization task under this constraint, a representation f_{proxy} is introduced. This proxy function is trained to mimic the functional behavior of f_{gated} as precisely as possible. The gated camera receives only the photons from scene S that are visible in the output image due to the possibility of gating. However, the proxy function receives a full gated image I in which the complete scene is illuminated as input and mimics the gating changing the brightness of the pixels according to the defined gated parameters. Because the proxy function f_{proxy} is realized by a fully convolutional network (FCN), the output O_{proxy} results from the dependencies between its learnable parameters Θ_p, the input image$I$, a to the scene corresponding LiDAR depth map L and the hyperparameters P:

$$O_{proxy} = f_{proxy}(I, L, P; \Theta_p)$$

The additional LiDAR-input is included to provide depth cues. The proxy function can be evaluated for every arbitrary combination of hyperparameters $P_n \in P$. Moreover, f_{proxy} is differentiable, so that $\partial f_{proxy} / \partial P_n$ can be acquired. During the training process, the learnable parameters Θ_p need to be adapted in such a way, that $O_{proxy} \approx O_{gated}$ for any combination of input image I and hyperparameters P. The LiDAR-data gives guidance for a better reproduction fgated. The adaption can be achieved by solving the following optimization problem that searches for optimal learnable parameters Θ_p :

$$\Theta_p \quad = argmin_{\{\Theta_p\}} \sum_{i=1}^{N} \sum_{j=1}^{M} L_{proxy}^{mult}\left(f_{proxy}(I_i, L_i, P_j; \Theta_P), f_{gated}(S_i, P_j)\right).$$

For training, M different parameter settings are used. The loss function L_{proxy}^{mult}measures the deviation between the proxy function and the gated camera output. Since $f_{proxy}$is differentiable, it can be minimized by applying a gradient-based optimization method.

Step 2: Optimizing Gated Parameters with Differentiable Proxies

By solving the final Equation of step 1 and obtaining the optimal learnable parameters Θ_p , we get the possibility to substitute f_{gated} with f_{proxy}. This results in the change of the initial search problem from to:

$$P \quad = arg\min_{\{P_1,P_2,P_3\}} \sum_{i=1}^{N} L_{depth}^{mult}\left(f_{depth}\left(O_{proxy}^{i,1}, O_{proxy}^{i,2}, O_{proxy}^{i,3}\right), T_i\right)$$

$$\text{with } O_{proxy}^{i,k} = f_{proxy}\left(I_i, L_i, P_k; \Theta_p\ \right) k \in \{1,2,3\}.$$

This creates a more approachable problem since the proxy function can be evaluated at any hyperparameter combination without the cumbersome data collection with the gated camera. In addition, there is now the possibility of optimizing the equation above with a derivative-based method such as stochastic gradient-descent since all included functions are derivable. So far, it has been assumed that the depth function can consistently provide optimal depth maps. However, since such a generally valid function does not exist, this is also implemented by a neural network. This depth network is additionally dependent on learnable parameters Θ_d. Like the proxy network, the depth network needs to be trained to fulfil the depth-prediction task for arbitrary input images reliably. This is done by solving the following optimization problem to search for the optimal learnable parameters Θ_d:

$$\Theta_d \ = argmin_{\{\Theta_d\}} \sum_{i=1}^{N} L_{depth}^{mult}\left(f_{depth}\left(O_{proxy}^{i,1}, O_{proxy}^{i,2}, O_{proxy}^{i,3}\right), T_i\right)$$

$$\text{with } O_{proxy}^{i,k} = f_{proxy}\left(I_i, L_i, P_k; \Theta_p\ \right) k \in \{1,2,3\}.$$

The loss function L_{depth}^{mult}measures the deviation between the output of the depth function O_{depth} and the ground truth depth maps that correspond to the input scene. The hyperparameters P_1, P_2, P_3 are fixed and take the non-optimized values proposed by Gruber et al. [57]. This process is referred to as pre-training the depth network. During the search for optimal parameters, the learnable parameters Θ_d that were optimized for the non-optimized parameters are optimized further to better adapt to the changing hyperparameters that determine the inputs for the depth-function. Now, the equation for the search of the optimal hyperparameters is denoted as:

$$P \quad , \Theta_d = arg\min_{\{P_1,P_2,P_3,\Theta_d\}} \sum_{i=1}^{N} L_{depth}^{mult}\left(f_{depth}\left(O_{proxy}^{i,1}, O_{proxy}^{i,2}, O_{proxy}^{i,3}\right), T_i\right)$$

$$\text{with } O_{proxy}^{i,k} = f_{proxy}\left(I_i, L_i, P_k; \Theta_p\ \right) k \in \{1,2,3\}.$$

The hyperparameters are initialized with the non-optimized hyperparameters proposed by Gruber et al. [57]. The learnable parameters Θ_d^* optimized for the changed hyperparameters are a byproduct of the optimization and not of primary interest.

4.4 Form and Parameterization

Form and Parameterization of Proxy Network

For the realization of the proxy network, a variant of the U-Net architecture [60] is chosen. It consists of a contracting path to capture and process the context of the input image and a symmetric expanding path that can restore the initial image size. The contracting path, also called encoder, consists of the repeated application of two 3x3 convolutions, each followed by a leaky Rectified Linear Unit (ReLU) and a 2x2 max pooling operation with stride 2 for down-sampling. At each down-sampling step, the number of feature channels is doubled. In the expansive path, also denoted as decoder, an up-sampling followed by a 2x2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the encoder, and two 3x3 convolutions followed by a leaky ReLU is performed. The used variant has {32,64,128,256} encoder channels and vice versa for the decoder. In addition to the full gated image and the LiDAR depth map, four more channels are provided as input. The additional channels contain the values of the gated parameters replicated over the spatial dimension. To encourage the model to learn the effects of the hyperparameters on the output image, the hyperparameter channels are further appended to each down-sample layer [30]. Each up- and down-sampling operation is performed independently per channel to avoid unwanted dependencies. The hyperparameter inputs, the LiDAR depth maps and the full gated image are normalized to lie in the [0,1] interval. This allows for a more stable training [30]. To provide better guidance during training, a multi-scale loss is used. Two additional stages of the decoder are extracted and flattened. The resulting images with a lower resolution than the output are then compared to a down-sampled version of the ground truth. The model is trained with the synthetic dataset.

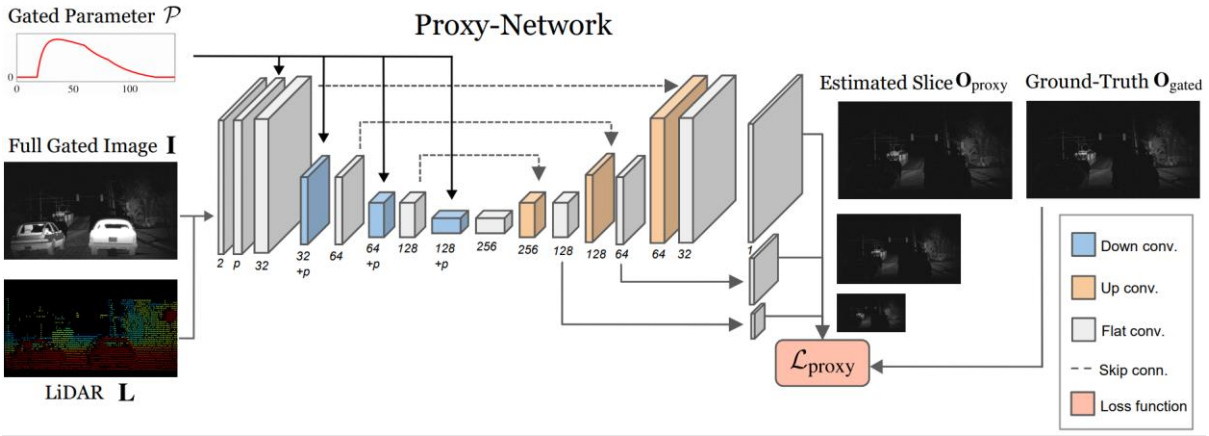

Figure 4.5: Architecture of the proxy network. The input of the network are p = 4 gated parameters, the full gated image and the LiDAR depth map. The proxy network is inspired by the U-Net architecture but consists of additional parameter appendages at every down convolution step. The loss is derived by comparing the output of the network with a ground truth gated slice in three scales.

Form and Parameterization of Depth Network

The depth network is realized using a variant of the U-Net architecture as well. This differs from the variant used for the proxy network in the following aspects: It receives only the three gated slices as inputs and

therefore has only three input channels. Also, nothing is appended to each down.-sample layer. In this case, a multi-scale loss is implemented as well. Following Gruber et al. [57], the target depth maps are limited to cover a range up to 150m. The gated slices and the target depth maps are normalized to lie in the [0, 1] interval for more stable training. The architecture of the network that fulfils the depth prediction is depicted in Figure 4.6 in the pre-training state. Figure 4.7 shows the complete architecture of the neural network used for searching for optimal hyperparameters. There, the proxy function is used for the creation of gated slices. Those slices are then processed to a dense depth map by the depth network.

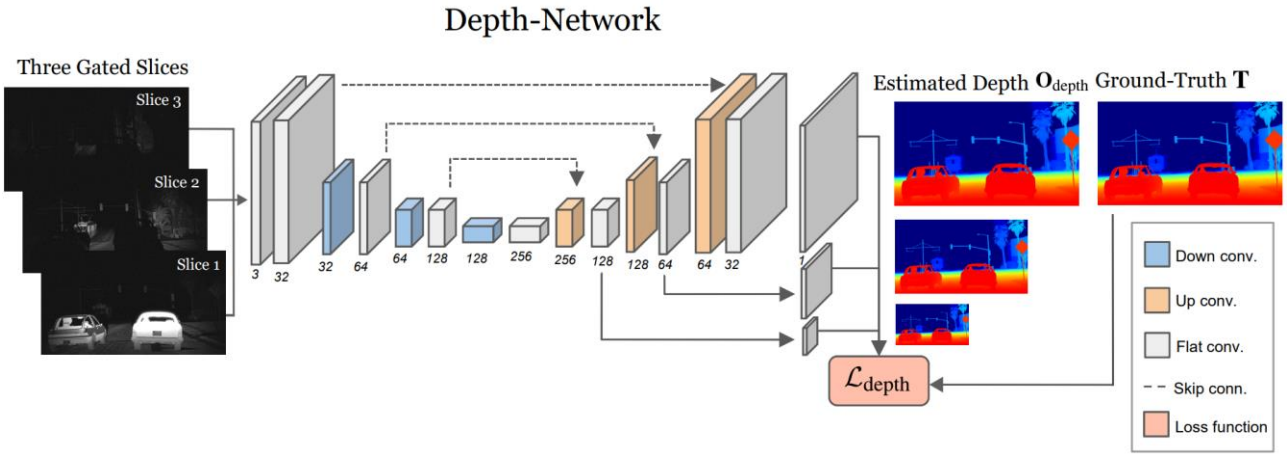

Figure 4.6: Architecture of the depth network. Three gated slices are used as input. The architecture is based on the U-Net. The loss is derived by comparing the output of the network with a ground truth depth map in three scales.

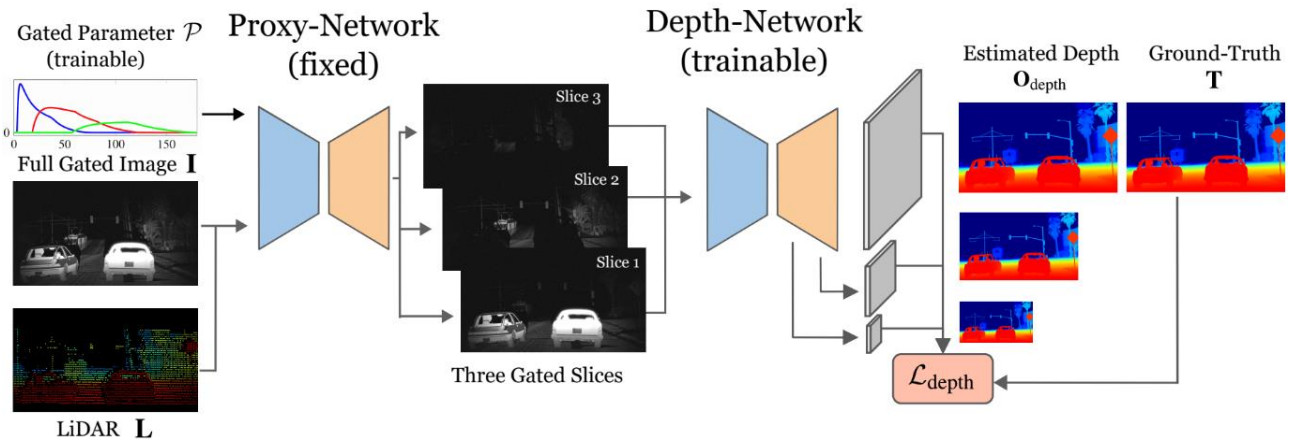

Figure 4.7: Architecture of the Optimization Network. The proxy network is used to predict three different gated slices for one scene, depending on the gated parameters. The three slices are then used to predict a depth map using the depth network. Since the proxy function is differentiable, an optimizer can be used for optimizing the gated parameters with respect to the depth-specific loss.

4.5 Loss

The loss function defines the objective which is optimized during training. Since the proxy function and the depth function don't fulfil the same task, different loss function was chosen. These are described in more detail below. First, an explanation to how the loss is derived in multiple stages is given.

Multi-Scale Loss

The multi-scale loss penalizes differences between the ground truth and the estimates at different scales that are extracted from the decoder of the FCN. The multi-scale loss is defined by

$$L_{task}^{mult}(y, y') = \sum_{i=1}^{s} \lambda_{m_i} L_{task}\left(y^{(i)}, y'^{(i)}\right),$$

where $y^{(i)}$and $y'^{(i)}$ are the network's output and target at a scale (i). $L_{task}\left(y^{(i)}, y'^{(i)}\right)$ is the task-specific loss at scale (i). The task refers to either the proxy network (L_{proxy}) or the depth network (L_{depth}). λ_{m_i} is the weight of the loss at the corresponding scale. As shown in Figure 4.5 and Figure 4.6, for both tasks $S = 3$ scales denoted as $i = \{0,1,2\}$ are defined. Scale 0 corresponds to the output of the network with the highest resolution. Scale 1 and 2 are from earlier stages with lower resolutions. The target images are resized to have a corresponding resolution of $1/2^i$. The weight of each scale is defined by

$$\lambda_{m_i} = 1 - 0.2i$$

This achieves a higher weight for an output with a higher resolution. In general, using a multiscale loss leads to better training behaviour since the network is guided to take the appearance of the desired target already at earlier stages.

Loss of Proxy Network

The loss function is the task-specific metric that is minimized by the NN during the training process. In the case of the proxy-function, the loss L_{proxy} is chosen to be a combination of an L_1term and a variation of the Structural Similarity (SSIM) index term, defined the following by Godard et al. [61]:

$$L_{proxy}(O_{proxy}, O_{gated}) = 0.85\frac{1 - SSIM(O_{proxy}, O_{gated})}{2} + 0.15\left|\left|O_{proxy} - O_{gated}\right|\right|_1$$

Loss of Depth Network

For pre-training the depth network, as well as optimizing the parameters, the reverse Huber (berHu) loss as introduced by Laina et al. [62] is used. It is defined by

$$L_{depth}(O_{depth}, T) = \frac{1}{N}\sum_{i=1}^{N} B(O_{depth,i} - T_i)$$

$$\text{with } B(x) = \begin{cases} |x| \vee x \vee \leq c \\ \frac{x^2 + c^2}{2c} |x| > c \end{cases}$$

The variable c is calculated with $c = \frac{1}{5}\underset{i}{max}\left(\left|O_{depth,i} - T_i\right|\right)$ where i indexes all N pixels over each image. That means that the boundary is set to 20% of the maximal L_1 deviation that is observed in the considered

images. The berHu loss corresponds to the L_1 norm when $x \in [-c, c]$ and to the L_2 norm outside this range. It puts high weight towards corresponding pixels with a high deviation because of the L_2 term. At the same time, L_1 accounts for a greater impact of smaller deviations than L_2 would [62].

4.6 Results

The optimal parameters were chosen to be those that can achieve the highest accuracy for the depth prediction task. These can be obtained when pre-training the depth network with a learning rate of 0.0001 for 10 epochs and optimizing the parameters afterwards for 30 epochs while training the depth network with a learning rate of 0.001 with learning rate decay both using the berHu loss. The optimized parameters can be found in Table 4.1.

Baseline

To determine the improvement for depth estimation with gated images by optimizing the gating parameters, a comparison to a baseline, which estimates depth from non-optimized gated slices, needs to be created. Since the goal is to show an improvement in depth estimation achieved solely by optimizing the gated parameters, a baseline is created with a FCN based method that uses the non-optimized parameters. It is trained in a supervised way, with the ground truth depth maps contained in the dataset. The inputs are generated with the synthetic dataset framework using the parameters proposed by Gruber et al. [57], shown in Table 4.1. The U-Net is used. The learning rate is set to 0.0001. The loss function is chosen to be the berHu loss and it is trained for 25 epochs. The results that are obtained are evaluated on the test dataset. The best results were achieved after 23 epochs.

Table 4.1: Baseline and optimized gating parameters

Parameters	Baseline Parameters			Optimized Parameters		
	Slice 1	Slice 2	Slice 3	Slice1	Slice2	Slice 3
Pulses	202	591	770	176	566	907
$t_0[ns]$	260	400	750	310.40	452.38	756.44
$t_g[ns]$	220	420	420	165.48	364.37	448.18
$t_l[ns]$	240	280	370	268.18	306.72	427.04

Qualitative Results

The depth maps in Figure 4.8 show a clear gain for the optimized parameters compared to the baseline parameters. The error maps indicate larger deviations from the ground truth with brighter colours. In most images, noise in the sky region can be reduced or eliminated ((a), (b), (d)). In addition, fine details can be reproduced better (d). In some samples, where the baseline parameters are able to provide good estimations, no visual improvement can be observed (c). This example shows also that sporadic spots can occur where the baseline parameters are able to predict the depth map with a lower deviation to the

ground truth. However, examples where the optimized parameters lead to an overall visibly inferior quality or images with systematic errors cannot be identified.

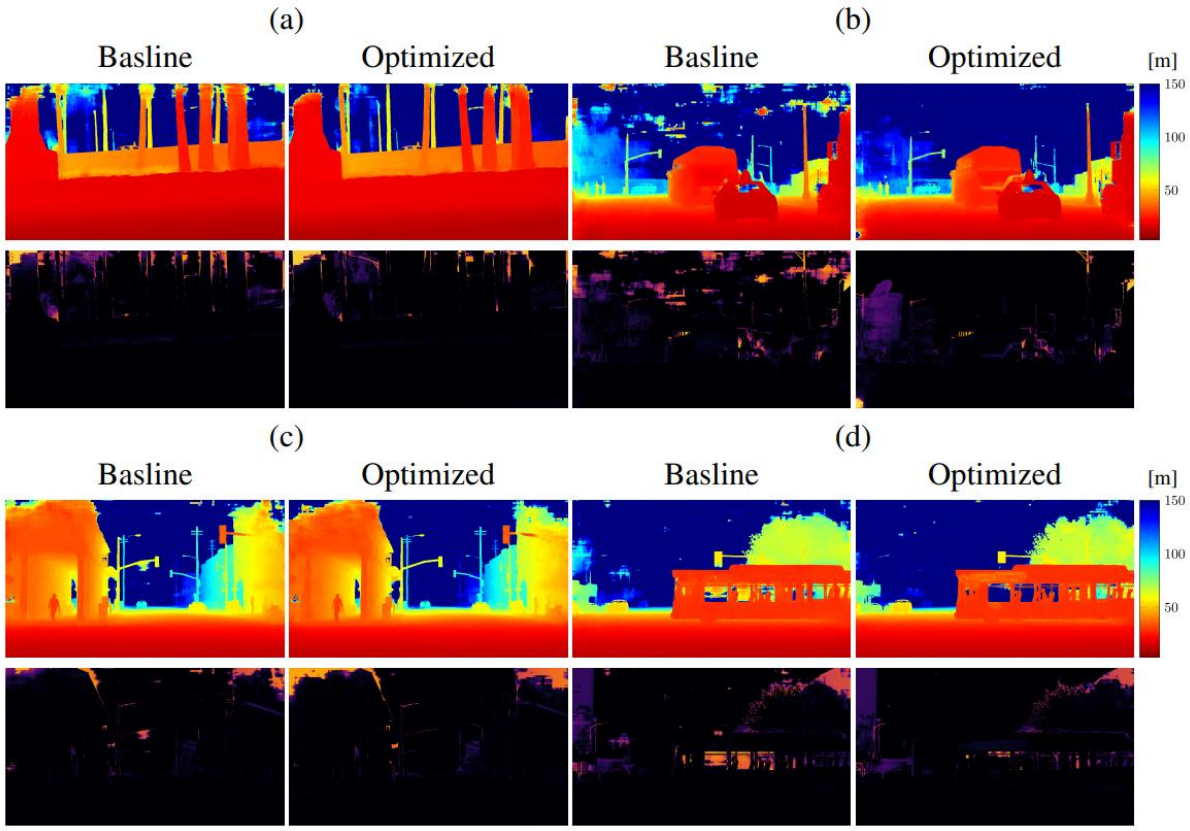

Figure 4.8: Comparison of baseline and optimized depth estimations. Below the depth estimation, the corresponding error map is displayed, where brighter areas indicate.

Quantitative Results

The metrics in Table 4.2, which are obtained by evaluating the test dataset, show a clear improvement for the optimized parameters. The MAE reached an improvement of 13.1% and the RMSE an improvement of 5.9%.

Table 4.2: Results of depth networks trained with baseline and optimized parameters.

Configuration	MAE	RMSE	δ_1	δ_2	δ_3
Baseline	3.555	12.811	95.815	97.656	98.410
Optimized	**3.090**	**12.050**	**96.059**	**97.979**	**98.621**

4.7 Conclusion

For this work, a framework for optimizing gated parameters with the use of proxy functions is implemented. An optimization for the task of reconstructing depth maps from three synthetically generated gated slices with FCNs is performed. The proxy function is implemented as a variant of the U-Net neural network. It is found that using a full gated image and LiDAR data as input leads to a model that can recreate gated images with sufficient accuracy. For optimizing the gated parameters, a depth network needs to be pre-trained with gated slices that are created with unoptimized parameters. Following, the gated parameters that determine the input images for the depth network, created with the proxy function can be optimized with a gradient based optimizer. It is important that both gated parameters and learnable parameters of the depth network are adaptable during the optimization process. The optimized parameters create gated images where the first and second slice have a lower intensity compared to handcrafted baseline images. The third slice is characterized by a higher intensity and a higher covered range. In a direct comparison between baseline and optimized parameters, a clear improvement in the created depth maps can be observed. Improvements are noticeable through reduced noise and a better reproduction of fine details. Thus, it can be concluded that the optimization of gated parameters is effective and has the potential to improve existing state-of-the-art methods for depth estimation from gated images.

Future Work

In a next step, the optimization of the gated parameters for depth estimation can be performed for real-world data. Here, the concepts that have proven to be effective with the use of synthetic data can be adopted. An optimization of gated parameters for other environmental conditions is also imaginable. Since LiDAR is unreliable in adverse weather, other inputs for the proxy functions need to be investigated. Possible options would be the use of three gated slices, stereo data or a full gated image. For the use of three gated slices, the influence of motion artefacts has to be minimized. When using stereo images, two gated cameras are necessary. A good model from full gated images can be achieved by using a larger dataset and more sophisticated neural network architectures. Additionally, the gated parameters could be optimized for other tasks like object detection. Here, optimization could for example make an important contribution in improving early detection of lost cargo.

List of abbreviations

Abbreviation	Meaning
APD	Avalanche Photo Diode
berHu	Reverse Huber
BRDF	Bidirectional Reflectance Distribution Function
CMA-ES	Covariance Matrix Adaptation-Evolution Strategy
CNN	Convolutional Neural Network
DSP	Digital Signal Processing
FCN	Fully Convolutional Network
GTAV	Grand Theft Auto V
IoU	Intersection-over-Union
ISP	Image Signal Processor
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
mAP	Mean of Average Precision
MEMS	Micro Electro Mechannical System
MOO	Multi-Objective Optimization
NIR	Near InfraRed
PC	Point Cloud
ReLU	Rectified Linear Unit
RMSE	Root-Mean-Square Error
SNR	Signal-to-Noise Ratio
SPAD	Single-Photon Avalanche Diode
SSIM	Structural Similarity

References

[1] 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, 2017, vol. 78, pp. 1–16. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[2] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, “PV-RCNN: Point-voxel feature set abstraction for 3D object detection,” in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020.

[3] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, “Motion-based detection and tracking in 3D LiDAR scans,” in *IEEE international conference on robotics and automation (ICRA)*, 2016. doi: 10.1109/ICRA.2016.7487649.

[4] J. Zhang, and S. Singh, “LOAM : LiDAR odometry and mapping in real-time,” *Robotics: Science and Systems Conference (RSS)*, 2014.

[5] 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,” in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020.

[6] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, “Argoverse: 3D tracking and forecasting with rich maps,” in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2019. doi: 10.1109/CVPR.2019.00895.

[7] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide, “Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather,” in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020.

[8] D. Wang, C. Watkins, and H. Xie, “MEMS mirrors for LiDAR: A review,” *Micromachines*, vol. 11, no. 5, 2020.

[9] F. Villa, B. Markovic, S. Bellisai, D. Bronzi, A. Tosi, F. Zappa, S. Tisa, D. Durini, S. Weyers, U. Paschen, and W. Brockherde, “SPAD smart pixel for time-of-flight and time-correlated single-photon counting measurements,” *IEEE Photonics Journal*, vol. 4, no. 3, pp. 795–804, 2012.

[10] G. Williams, “Optimization of eyesafe avalanche photodiode LiDAR for automobile safety and autonomous navigation systems,” *Optical Engineering*, vol. 56, no. 3, pp. 1–9, 2017, doi: 10.1117/1.OE.56.3.031224.

[11] M. Bijelic, T. Gruber, and W. Ritter, “A benchmark for LiDAR sensors in fog: Is detection breaking down?” in *IEEE intelligent vehicles symposium (IV)*, 2018. doi: 10.1109/IVS.2018.8500543.

[12] A. Carballo, J. Lambert, A. Monrroy-Cano, D. R. Wong, P. Narksri, Y. Kitsukawa, E. Takeuchi, S. Kato, and K. Takeda, “LIBRE: The multiple 3D LiDAR dataset,” in *IEEE intelligent vehicles symposium (IV)*, 2020. doi: 10.1109/IV47402.2020.9304681.

[13] M. Hahner, C. Sakaridis, D. Dai, and L. Van Gool, “Fog simulation on real LiDAR point clouds for 3D object detection in adverse weather,” in *IEEE international conference on computer vision (ICCV)*, 2021.

[14] R. Heinzler, P. Schindler, J. Seekircher, W. Ritter, and W. Stork, “Weather influence and classification with automotive LiDAR sensors,” in *IEEE intelligent vehicles symposium (IV)*, 2019. doi: 10.1109/IVS.2019.8814205.

[15] M. Jokela, M. Kutila, and P. Pyykönen, “Testing and validation of automotive point-cloud sensors in adverse weather conditions,” *Applied Sciences*, vol. 9, 2019, doi: 10.3390/app9112341.

[16] A. Wallace, A. Halimi, and G. Buller, “Full waveform LiDAR for adverse weather conditions,” *IEEE Transactions on Vehicular Technology*, vol. 69, 2020, doi: 10.1109/TVT.2020.2989148.

[17] C. Goodin, D. Carruth, M. Doude, and C. Hudson, “Predicting the influence of rain on LiDAR in ADAS,” *Electronics*, vol. 8, 2019, doi: 10.3390/electronics8010089.

[18] M. Kutila, P. Pyykönen, M. Jokela, T. Gruber, M. Bijelic, and W. Ritter, “Benchmarking automotive LiDAR performance in arctic conditions,” in *IEEE international conference on intelligent transportation systems (ITSC)*, 2020.

[19] S. Michaud, J.-F. Lalonde, and P. Giguère, “Towards characterizing the behavior of LiDARs in snowy conditions,” in *Workshop on planning, perception and navigation for intelligent vehicles, IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2015.

[20] M. Hahner, C. Sakaridis, M. Bijelic, F. Heide, F. Yu, D. Dai, and L. Van Gool, “LiDAR Snowfall Simulation for Robust 3D Object Detection,” in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

[21] V. Kilic, D. Hegde, V. Sindagi, A. B. Cooper, M. Foster, and V. Patel, “LiDAR light scattering augmentation (LISA): Physics-based simulation of adverse weather conditions for 3D object detection,” *arXiv preprint 2107.07004*, 2021.

[22] M. Tremblay, S. S. Halder, R. de Charette, and J.-F. Lalonde, “Rain rendering for evaluating and improving robustness to bad weather,” *International Journal of Computer Vision (IJCV)*, vol. 126, 2021.

[23] C. Sakaridis, D. Dai, and L. Van Gool, “Semantic foggy scene understanding with synthetic data,” *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, 2018.

[24] T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari, and F. Yu, “SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022, pp. 21371–21382.

[25] X. Weng, Y. Man, D. Cheng, J. Park, M. O’Toole, and K. Kitani, “All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds,” *arXiv*, 2021.

[26] B. Hurl, K. Czarnecki, and S. Waslander, “Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception.” arXiv, May 2019. doi: 10.48550/arXiv.1905.00160.

[27] N. Hansen, “The CMA evolution strategy: A tutorial.” arXiv, 2016. doi: 10.48550/ARXIV.1604.00772.

[28] A. Mosleh, A. Sharma, E. Onzon, F. Mannan, N. Robidoux, and F. Heide, “Hardware-in-the-Loop End-to-End Optimization of Camera Image Processing Pipelines,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 7526–7535. doi: 10.1109/CVPR42600.2020.00755.

[29] N. Robidoux, L. E. G. Capel, D. Seo, A. Sharma, F. Ariza, and F. Heide, “End-to-End High Dynamic Range Camera Pipeline Optimization,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6297–6307. Accessed: Apr. 26, 2022. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Robidoux_End-to-End_High_Dynamic_Range_Camera_Pipeline_Optimization_CVPR_2021_paper.html

[30] E. Tseng, F. Yu, Y. Yang, F. Mannan, K. ST. Arnaud, D. Nowrouzezahrai, J. F. Lalonde, and F. Heide, “Hyperparameter optimization in black-box image processing using differentiable proxies,” *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 1–14, Aug. 2019, doi: 10.1145/3306346.3322996.

[31] E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J. F. Lalonde, and F. Heide, “Differentiable compound optics and processing pipeline optimization for end-to-end camera design,” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 2, pp. 1–19, 2021.

[32] J. Nishimura, T. Gerasimow, S. Rao, A. Sutic, C.-T. Wu, and G. Michael, “Automatic ISP image quality tuning using non-linear optimization.” arXiv, Feb. 2019. Accessed: Oct. 06, 2022. [Online]. Available: http://arxiv.org/abs/1902.09023

[33] A. Mittal, A. K. Moorthy, and A. C. Bovik, “Automatic parameter prediction for image denoising algorithms using perceptual quality features,” in *Human Vision and Electronic Imaging XVII*, Feb. 2012, vol. 8291, pp. 110–116. doi: 10.1117/12.912243.

[34] L. Pfister and Y. Bresler, “Learning Filter Bank Sparsifying Transforms,” *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 504–519, Jan. 2019, doi: 10.1109/TSP.2018.2883021.

[35] B. Behroozpour, P. A. M. Sandborn, M. C. Wu, and B. E. Boser, “Lidar System Architectures and Circuits,” *IEEE Communications Magazine*, vol. 55, no. 10, pp. 135–142, Oct. 2017, doi: 10.1109/MCOM.2017.1700030.

[36] R. Rasshofer, M. Spies, and H. Spies, “Influences of weather phenomena on automotive laser radar systems,” *Advances in Radio Science*, vol. 9, 2011, doi: 10.5194/ars-9-49-2011.

[37] I. Maksymova, C. Steger, and N. Druml, “Review of LiDAR Sensor Data Acquisition and Compression for Automotive Applications,” *Proceedings*, vol. 2, no. 13, p. 852, 2018, doi: 10.3390/proceedings2130852.

[38] A. K. Pediredla, A. C. Sankaranarayanan, M. Buttafava, A. Tosi, and A. Veeraraghavan, “Signal Processing Based Pile-up Compensation for Gated Single-Photon Avalanche Diodes.” arXiv, Jun. 2018. Accessed: Sep. 12, 2022. [Online]. Available: http://arxiv.org/abs/1806.07437

[39] F. Pittaluga, Z. Tasneem, J. Folden, B. Tilmon, A. Chakrabarti, and S. Koppal, “Towards a MEMS-based adaptive LIDAR,” Oct. 2020.

[40] K. Nakamura, K. Narumi, K. Kikuchi, and Y. Inada, “Liquid crystal-tunable optical phased array for LiDAR applications,” in *Smart Photonic and Optoelectronic Integrated Circuits XXIII*, Mar. 2021, vol. 11690, pp. 94–99. doi: 10.1117/12.2591230.

[41] N. Vödisch, O. Unal, K. Li, L. Van Gool, and D. Dai, “End-to-End Optimization of LiDAR Beam Configuration for 3D Object Detection and Localization,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2242–2249, Apr. 2022, doi: 10.1109/LRA.2022.3142738.

[42] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, “Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications.” Mar. 2018. doi: 10.1007/s11263-018-1073-7.

[43] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to End Learning for Self-Driving Cars.” arXiv, Apr. 2016. Accessed: Oct. 06, 2022. [Online]. Available: http://arxiv.org/abs/1604.07316

[44] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles.” arXiv, Jul. 2017. doi: 10.48550/arXiv.1705.05065.

[45] S. R. Richter, Z. Hayder, and V. Koltun, “Playing for Benchmarks.” arXiv, Sep. 2017. doi: 10.48550/arXiv.1709.07322.

[46] Y. Man, X. Weng, P. K. Sivakumar, M. O’Toole, and K. Kitani, “Multi-echo LiDAR for 3D object detection,” in *2021 IEEE/CVF international conference on computer vision (ICCV)*, 2021, pp. 3743–3752. doi: 10.1109/ICCV48922.2021.00374.

[47] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 3234–3243. doi: 10.1109/CVPR.2016.352.

[48] D. Shin, “Computational imaging with small numbers of photons,” Thesis, Massachusetts Institute of Technology, 2016. Accessed: Oct. 11, 2022. [Online]. Available: https://dspace.mit.edu/handle/1721.1/103743

[49] M. O’Toole, F. Heide, D. B. Lindell, K. Zang, S. Diamond, and G. Wetzstein, “Reconstructing Transient Images from Single-Photon Sensors,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2289–2297. doi: 10.1109/CVPR.2017.246.

[50] F. Heide, S. Diamond, D. B. Lindell, and G. Wetzstein, “Sub-picosecond photon-efficient 3D imaging using single-photon sensors,” *Scientific reports*, vol. 8, no. 1, pp. 1–8, 2018.

[51] R. L. Cook and K. E. Torrance, “A reflectance model for computer graphics,” *ACM SIGGRAPH Computer Graphics*, vol. 15, no. 3, pp. 307–316, Aug. 1981, doi: 10.1145/965161.806819.

[52] X. Li, B. Yang, X. Xie, D. Li, and L. Xu, “Influence of Waveform Characteristics on LiDAR Ranging Accuracy and Precision,” *Sensors*, vol. 18, no. 4, p. 1156, Apr. 2018, doi: 10.3390/s18041156.

[53] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, “Combining convergence and diversity in evolutionary multiobjective optimization,” *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[54] M. T. Emmerich and A. H. Deutz, “A tutorial on multiobjective optimization: Fundamentals and evolutionary methods,” vol. 17, no. 3, pp. 585–609, Sep. 2018, doi: 10.1007/s11047-018-9685-y.

[55] D. V. Arnold and N. Hansen, “Active covariance matrix adaptation for the (1+1)-CMA-ES,” in *Proceedings of the 12th annual conference on genetic and evolutionary computation*, 2010, pp. 385–392.

[56] K. Nishida and Y. Akimoto, “Population size adaptation for the CMA-ES based on the estimation accuracy of the natural gradient,” Jul. 2016. doi: 10.1145/2908812.2908864.

[57] T. Gruber, F. Julca-Aguilar, M. Bijelic, W. Ritter, K. Dietmayer and F. Heide, *Gated2Depth: Real-time Dense Lidar from Gated Images*, 2019, p. 11.

[58] T. Gruber, “Real-Time Super-Resolved Depth Estimation for Self-Driving Cars from Multiple Gated Images,” *Open Access Repositorium der Universität Ulm und Technischen Hochschule Ulm,* 2020.

[59] Rockstar Games. Grand Theft Auto V. [Online], Available: https://www.rockstargames.com/gta-v [Accessed 23 11 2022].

[60] O. Ronneberger, P. Fischer und T. Brox, „U-Net: Convolutional Networks for Biomedical Image Segmentation,“ in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Bd. 9351, Cham, Springer International Publishing, 2015, p. 234–241.

[61] C. Godard, O. M. Aodha und G. J. Brostow, „Unsupervised Monocular Depth Estimation with Left-Right Consistency,“ IEEE, 2017, p. 6602–6611.

[62] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari und N. Navab, *Deeper Depth Prediction with Fully Convolutional Residual Networks,* p. 12.