LiDAR-in-the-loop Hyperparameter Optimization Supplementary Material

Félix Goudreault[1] Dominik Scheuble[2] Mario Bijelic[2,3]
Nicolas Robidoux[1] Felix Heide[1,3]
[1]Algolux [2]Mercedes-Benz [3]Princeton University

In this supplemental document, we describe the experimental setups in detail and show additional experimental results in support of the findings from the main manuscript (Section 1); we further explain the proposed LiDAR simulation method as well as the DSP pipeline (Section 2); and, finally, we review the concept of Pareto optimality and present the proposed optimization method in full (Section 3).

Contents

1. Additional Experimental Details and Results

1.1. Initial Settings for Optimization

In all of this work's simulated optimization experiments, the initial hyperparameter value was invariably the expert-tuned one. We did not investigate stability with respect to changes in the initial conditions. Note however that losses are noisy; this is briefly discussed in Section 1.4. Loss noisiness injects stochasticity in the optimization.

The simulated LiDAR was manually optimized by grid sampling and this expert-tuned setting was used to train the object detector (Section 1.7). This process revealed that most settings do not provide usable data, demonstrating the tedious nature of manual tuning and further justifying our automated hyperparameter optimization approach.

Finally, for the hardware LiDAR experiment, the expert-tuned parameters were vendor-optimized settings and the initial settings were the ones used to define the ground truth histogram (see Section 1.9).

1.2. LiDAR Depth and Intensity Objectives

Before presenting additional optimization results, we review the point cloud loss functions. We minimize the average RMS depth error and the average RMS intensity error, that is,

$$\mathcal{L}_{\text{depth}}(\boldsymbol{\Theta}) = \frac{1}{\mathrm{F}} \sum_{f=1}^{\mathrm{F}} \text{RMSE}(R_f, \Phi_f^{(R)}(\boldsymbol{\Theta})), \text{ and} \tag{1}$$

$$\mathcal{L}_{\text{int.}}(\boldsymbol{\Theta}) = \frac{1}{\mathrm{F}} \sum_{f=1}^{\mathrm{F}} \text{RMSE}(I_f, \Phi_f^{(I)}(\boldsymbol{\Theta})), \tag{2}$$

where F is the number of frames in the validation set. For frame f, the $\text{RMSE}(R_f, \Phi_f^{(R)}(\boldsymbol{\Theta}))$ is the pointwise root mean squared error between the predicted depth $\Phi_f^{(R)}(\boldsymbol{\Theta})$ and its corresponding ground truth R_f. Similarly, the $\text{RMSE}(I_f, \Phi_f^{(I)}(\boldsymbol{\Theta}))$ is the pointwise root mean squared error between the predicted integrated peak intensity $\Phi_f^{(I)}(\boldsymbol{\Theta})$ and corresponding ground truth I_f. We refer to Section 2.6 for the details of the computations of $\Phi_f^{(R,I)}(\boldsymbol{\Theta})$, R_f and I_f with our sensing and DSP model. Note that points that would have been marked as "missed" in either the predicted values or ground truth, resulting for example from a beam emitted towards the sky, are considered to have a distance and intensity of 0. The only stochastic component of the loss comes from the modeling of Poisson noise (Eq. (32)). A discussion of its impact on Pareto front computation is found in Section 1.4.

Computation Time. Loss evaluation largely dominates optimization run times. On a 6-GPU machine, one depth and intensity optimization run with 3000 loss evaluations requires approx. 6h to converge, four times that for runs with 12,000 loss evaluations. When object detection and classification is included, $10\times$ as many GPUs are required to match these run times. The high cost of evaluating losses is one reason why short run results—the runs with 3000 loss evaluations documented below—are the most important.

1.3. Multi-Objective Optimization of Depth and Intensity Average RMSE

We provide additional rendered results for the depth and intensity MOO experiment presented in Figure 4 of the main document. In Figure 1, we show the individual depth error per point encoded by

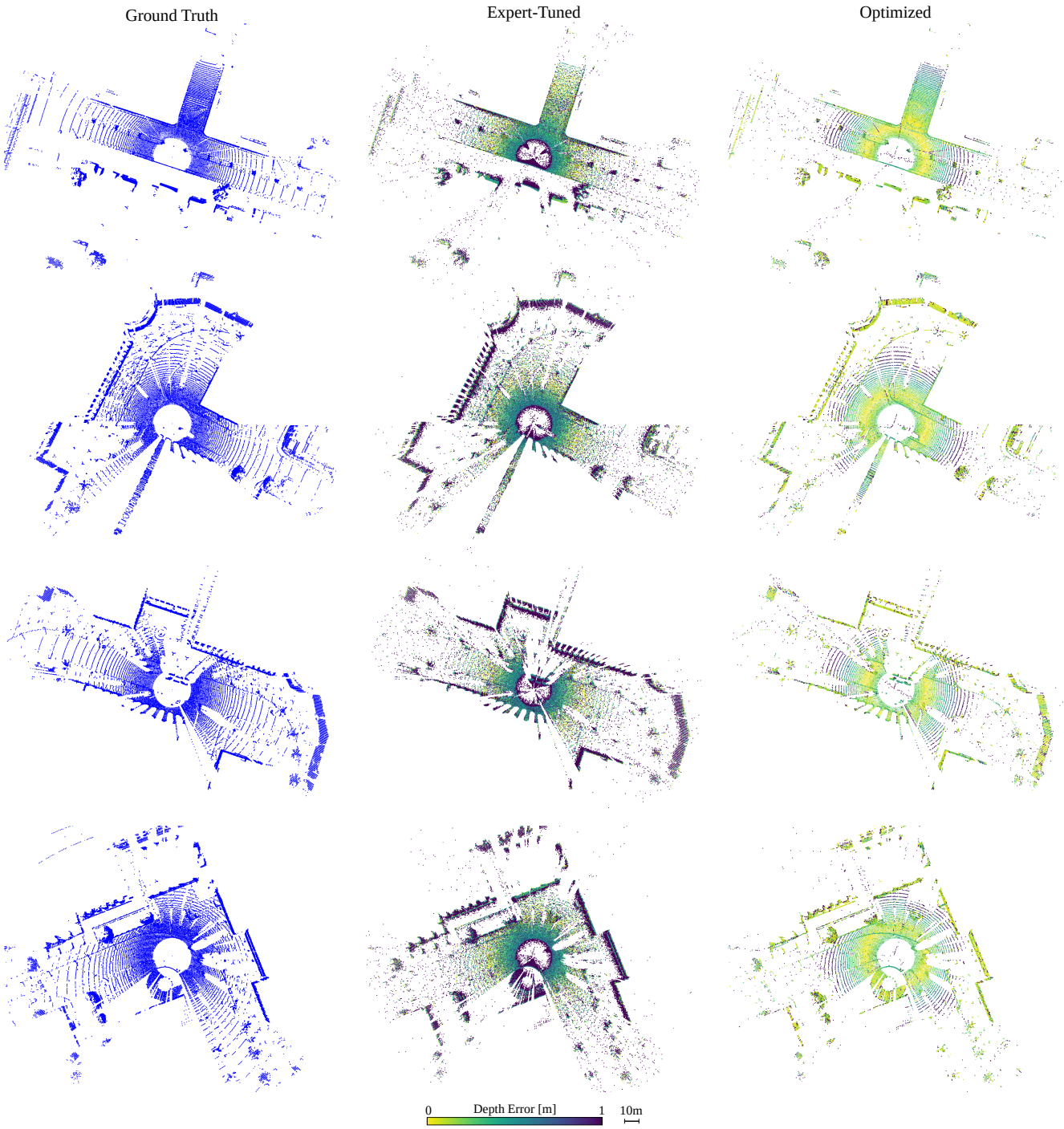

Figure 1. Comparison of expert-tuned and MOO optimized (for depth and intensity average RMSE) point clouds with ground truth in a bird's-eye view. We encode the individual depth error per point with color. Including depth as an optimization objective helps to remove clutter and more accurately resolve structures like facades of buildings.

color. The proposed method achieves significantly lower depth error than the expert-tuned one over the full range. Fine structures, especially, like the facades of buildings, are better resolved.

Validation Set. For all depth and intensity optimization runs, the validation set was composed of 10 frames randomly selected from the ones used for the object detector experiment. See Sec.1.7 for more details.

1.4. The ℓ_1-Norm as Evaluation Metric

As is well known, the ℓ_1-norm is invariant with respect to equal but opposite changes in pairs of non-negative vector components. We use the ℓ_1-norm of the loss vector as evaluation metric because the variations of the two loss components $\mathcal{L}_{\text{depth}}$ and $\mathcal{L}_{\text{int.}}$ are approximately balanced in the case of study, a property which the computation of the Pareto front (see Section 3.1 for a definition) revealed even though there was no reason to suspect it a priori. Nothing about the loss components' definitions—Equations (1)–(2) above—suggests that it should hold along the Pareto front; PC depth and PC intensity do not even have matching units! The Pareto front for the two losses is shown in the figure that accompanies Table 1 of the main paper. One readily sees that the Pareto front approximately follows a straight line with a slope of -1 which shows that the variations of the two losses are commensurate along the Pareto front.

Note that the Pareto front was estimated by aggregating loss data from all experiments and removing Pareto-dominated points. Computing an approximation of the Pareto front by removing dominated points from a limited number of samples ignores loss noisiness. We use a stochastic model for our experiments (sensor noise is modeled randomly, see Eq. (32)). However, loss noisiness appears to be relatively small. In Table 2 below, the difference Δ between the largest and smallest of three independent loss measurements obtained with expert-tuned LiDAR hyperparameter values, that is, using the same LiDAR configuration every time, is 0.006 for $\mathcal{L}_{\text{depth}}$, this loss component ranging from 8.900 to 12.186 for the reported hyperparameter values, and 0.014 for $\mathcal{L}_{\text{int.}}$, loss component which ranges between 1.565 and 5.016 in the table. Similar numbers can be extracted from Table 5, table which documents separate experiments. We infer that loss noise is small compared to loss variations in our case of study and expect that the main effect, given the large number of loss value samples—hundreds of thousands—which we used to compute the Pareto front, is that most of them are samples which the noise pushed slightly in the "right" direction.

In any case, the ℓ_1-norm, that is, the convex combination scalarization with unit weights, is generally not recommended for optimization or champion selection because *the ℓ_1-norm is not invariant with respect to rescalings of individual loss components*. Other scalarizations which *are* scale invariant like the stable max-rank loss defined in the main paper, and the (standard deviation) normalized ℓ_1-norm and the stable ℓ_1-rank defined in Section 1.6, are likely to be more robust for optimization and champion selection. Given that in our case of study the variations of two loss components *are* approximately balanced, the ℓ_1-norm, a simple metric which is distinct and separate from both the scalarizations that drive the proposed optimization method on the one hand, and the proposed champion selection method on the other, was chosen as primary evaluation metric.

In summary, although one should not rely on it without prior information about the respective variations ("gradients") of loss components, we use the ℓ_1-norm of the loss vector $\boldsymbol{\mathcal{L}} = (\mathcal{L}_{\text{depth}}, \mathcal{L}_{\text{int.}})$ as evaluation metric because our experiments showed that, in our specific case of study, the loss components are approximately balanced.

OBJECTIVE	$\mathcal{L}_{\text{depth}}$ ($\downarrow$)	$\mathcal{L}_{\text{int.}}$ ($\downarrow$)	ℓ_1-norm ($\downarrow$)
SOO on $\mathcal{L}_{\text{depth}}$	**8.870**	4.925	**13.795**
Expert-Tuned	12.115	1.976	14.091
SOO on $\mathcal{L}_{\text{int.}}$	30.979	**0.236**	31.215

Table 1. LiDAR $\mathcal{L}_{\text{depth}}$ SOO (top), and $\mathcal{L}_{\text{int.}}$ SOO (bottom), with corresponding value of the other loss obtained by direct evaluation of the loss component with the optimizing hyperparameter vector. The ℓ_1-norm of the loss vector $\left(\mathcal{L}_{\text{depth}}, \mathcal{L}_{\text{int.}}\right)$, that is, $\|\boldsymbol{\mathcal{L}}\|_1 = \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{int.}}$, which was (of course) not used to optimize, is shown in the third column. The best value for each metric is in **bold**.

1.5. Single Objective Vs. Multi-Objective Optimization

As explained in Section 3.2 below, hyperparameter vectors that minimize one loss generally make one or more of the other losses take relatively large values and consequently are unlikely to be balanced solutions of a genuinely multi-objective optimization (MOO) problem. One may nonetheless ask what would happen if the problem was treated as a single optimization problem with respect to one or the other of the two loss components. To answer this question, we performed two additional SOO experiments with the proposed CMA-ES variant[1], each one optimizing with respect to only one of the two loss components but also reporting the minimizing hyperparameter configuration's value of the other loss component. The results are shown in Table 1. They support the idea that optimizing for only one loss component yields solutions for which the other loss component's values are poor. Indeed, SOO driven by $\mathcal{L}_{\text{depth}}$ gets the best (smallest) value for this loss and the worst (largest) value for $\mathcal{L}_{\text{int.}}$ (compare with the values shown in Table 2 and Table 5, keeping in mind that losses are noisy and that solvers' results may, and often do, vary), and vice versa. In both cases, in fact, the loss component ignored by the optimizer has a worse value—a lot worse, in the case of SOO driven by $\mathcal{L}_{\text{int.}}$—than the expert-tuned configuration.

Another SOO approach is to use a "traditional" (static) scalarization of the loss vector [12], the minimizer of which is chosen as champion. Depending of the scalarization and solver, the optimized solution may of course vary considerably. In the following, we use this approach with a scalarization favorable in our case of study, namely the ℓ_1-norm of the loss vector (see Section 1.4). Three solvers were tested: the proposed solver (the CMA-ES variant shown in Algorithm 1; see Section 3.3), the CMA-ES variant of Mosleh *et al*. [34], and HyperOpt [3, 4]. Three independent optimization runs, with different random seeds, were performed with each solver. Results are shown in Table 2 and Figure 2.

First, let us compare with the single loss component SOO optimization results shown in Table 1. One sees that, with the proposed solver, $(\mathcal{L}_{\text{depth}}, \mathcal{L}_{\text{int.}})$ pairs are a lot more balanced, in that none of the values of the two loss components is relatively large. Indeed, with one single exception (the highest value of $\mathcal{L}_{\text{int.}}$ with 3000 loss evaluations), the proposed solver yields $\mathcal{L}_{\text{depth}}$ and $\mathcal{L}_{\text{int.}}$ values which are not only better balanced but that are also *simultaneously* better than those obtained with the expert-tuned hyperparameter settings. In other words, the expert-tuned setting is Pareto-dominated which is not the case with the other two solvers. In addition, with all three solvers, the ℓ_1-norm of the optimized loss vector is always[2] smaller than the expert-tuned one. This is expected as it drove optimization.

[1] When there is only one loss component, max-rank loss optimization with CMA-ES is strictly equivalent to optimizing directly with the loss because single objective CMA-ES is, by construction, invariant with respect to arbitrary strictly monotone remappings of the loss [21]. Rank is a monotone remapping.

[2] One exception arises if one compares HyperOpt's largest reported ℓ_1-norm with the expert-tuned result reported in

SOLVER	Steps	ℓ_1-norm (↓) min	ℓ_1-norm (↓) max	$\mathcal{L}_{\text{depth}}$ (↓) Low	$\mathcal{L}_{\text{depth}}$ (↓) Mid	$\mathcal{L}_{\text{depth}}$ (↓) High	$\mathcal{L}_{\text{int.}}$ (↓) High	$\mathcal{L}_{\text{int.}}$ (↓) Mid	$\mathcal{L}_{\text{int.}}$ (↓) Low	$\Delta\mathcal{L}_{\text{depth}}$ (↓)	$\Delta\mathcal{L}_{\text{int.}}$ (↓)
Expert-Tuned	1	14.124	14.143	12.180	12.182	12.186	1.957	1.944	1.943	0.006	0.014
Proposed	3000	**11.427**	**12.576**	9.861	**9.890**	**9.896**	2.686	**1.569**	**1.565**	**0.035**	1.121
	6000	**11.418**	**11.461**	9.850	**9.893**	**9.894**	**1.568**	**1.567**	**1.565**	**0.044**	**0.003**
	12000	**11.386**	**11.396**	9.850	**9.871**	**9.893**	**1.568**	**1.568**	**1.567**	**0.043**	**0.001**
Mosleh *et al.* [34]	3000	13.548	14.089	**8.900**	10.080	12.158	5.016	3.494	1.942	3.258	3.074
	6000	13.438	13.661	**8.987**	9.916	10.963	4.563	3.532	2.820	1.976	1.743
	12000	12.777	13.687	**9.001**	9.908	10.880	3.775	3.500	2.807	1.879	0.969
HyperOpt [3,4]	3000	12.232	14.122	10.033	12.168	12.169	**2.199**	1.953	1.939	2.136	**0.260**
	6000	-	-	-	-	-	-	-	-	-	-
	12000	-	-	-	-	-	-	-	-	-	-

Table 2. LiDAR SOO driven by the ℓ_1-norm $\|\boldsymbol{\mathcal{L}}\|_1$ of the loss vector $\boldsymbol{\mathcal{L}} = (\mathcal{L}_{\text{depth}}, \mathcal{L}_{\text{int.}})$ after 3000, 6000 and 12,000 loss evaluations. Three runs per solver. Along with the ℓ_1-norm itself, extreme and median values of the two loss components are shown, as well as the difference Δ between the highest and lowest. Note that an individual loss component may increase as the driving ℓ_1-norm decreases if the other loss component decreases more. **Bold** denotes the best solver for a given metric and total number of loss evaluations. Time constraints unfortunately limited the number of loss evaluations per run with the HyperOpt solver to 3000.

Now, let us compare Table 2 with the MOO results shown in Table 3. For each solver, Table 3 shows the median value, over three independent optimization runs, of the ℓ_1-norm of the champion selected using various methods. The last column, which shows the median value of the ℓ_1-norm minimizer over the three runs, is particularly relevant: It suggests that *MOO optimization generally finds better ℓ_1-norm of the loss than direct SOO driven by the ℓ_1-norm of the loss.* The min and max values over the three runs, shown in Table 4, basically tell the same story. This is quite striking given that our MOO experiments are *not* driven by the ℓ_1-norm of the loss vector. Although this could be a consequence of the tested SOO solvers not performing well, these results suggest that *MOO is better at finding synergies between loss components, synergies which are missed when the solver does not keep loss components separate.*

The results shown in Table 2 also suggest that the proposed solver finds better solutions (see the first column of the table) and is more stable than Mosleh *et al.*'s when used for SOO (see the last column). HyperOpt also does not perform as well as the proposed solver, but this conclusion is more fragile given the limited data. This being said, HyperOpt does not improve on the initial (based on the expert-tuned hyperparameters) loss value in two of the three runs; see the convergence curves shown in Figure 2. Hence, HyperOpt might be a delicate choice when optimizing problems with a limited number of optimizing steps, *e.g.*, when the cost of evaluating losses is high.[3]

1.6. Quality and Stability of MOO Solutions

The results shown in Table 1 of the main document come from a large study involving, for each solver, three different runs with different random seeds and 12,000 loss evaluations ("steps"). The results of this study are spread over Tables 3, 4, 5, and 6, and Figures 3, 4 and 5.

First, we note that, as seen in Tables 3, 4 and 5, MOO optimization almost invariably beats expert-

Table 1 instead of Table 2. This "failure", however, falls within the bounds of expected loss noisiness.

[3] After all, HyperOpt was designed for more general data structures than numerical parameter values in an hypercube [3,4].

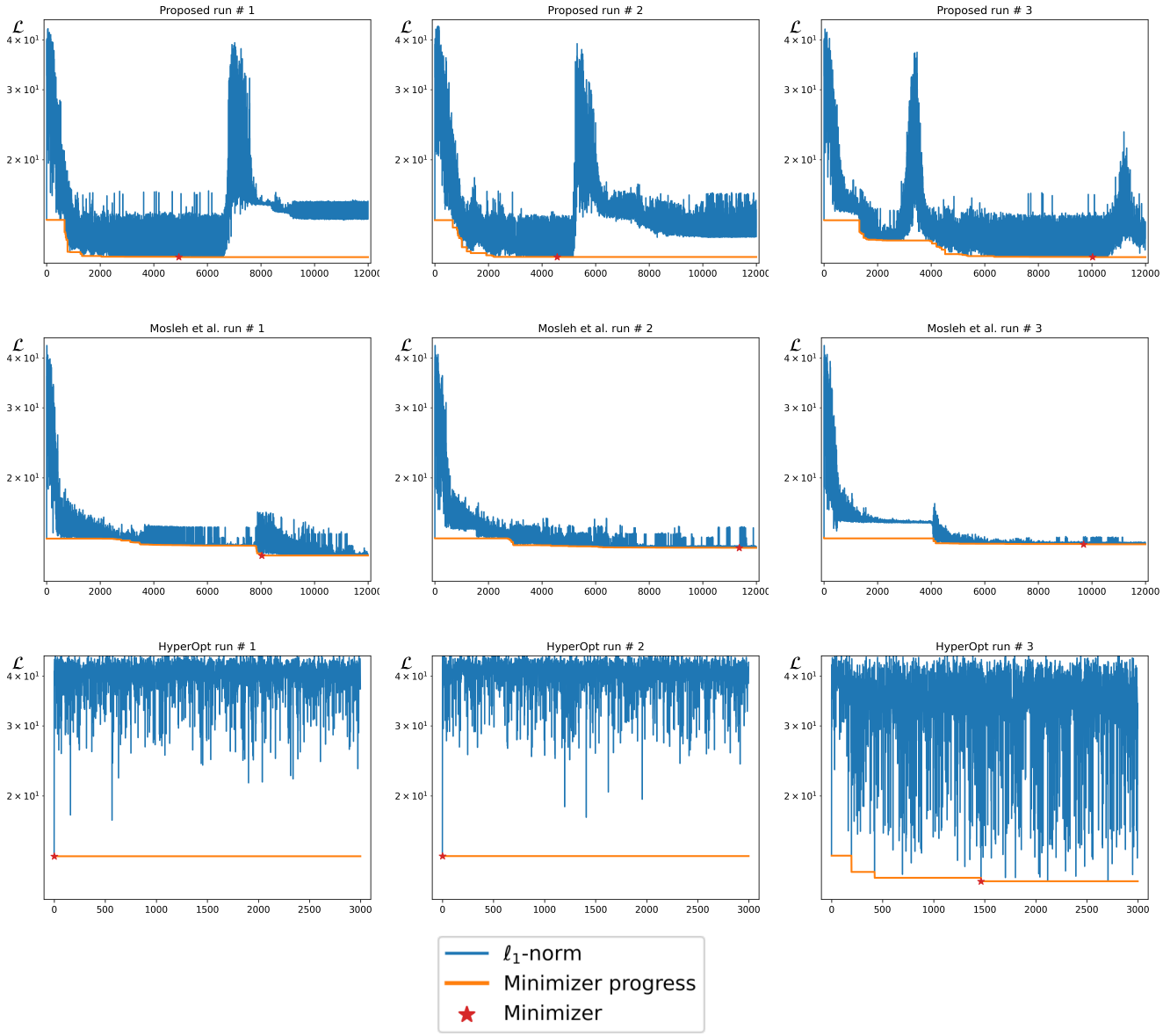

Figure 2. SOO convergence plots where the optimized objective was the ℓ_1-norm of loss vectors for the proposed, Mosleh *et al*. [34] and HyperOpt [3, 4] solvers. The blue curves correspond to the loss value while the orange curve tracks the ℓ_1-norm minimizer after each step. The red star denotes where the best minimizer of the run was found Unfortunately, time constraints limited the number of loss evaluations with the HyperOpt solver to 3000.

tuning.

This work's study, however, also addresses other questions.

As explained in Section 3.2, MOO generally yields a Pareto front that contains many solutions; for this reason, it is desirable to provide users with a default choice, with a *champion*. Mosleh *et al*. [34] and Robidoux *et al*. [43] take the last Pareto point of the run. As discussed in the main document and in Section 3.2, we propose a different champion selection criterion: Choose the Pareto point with the smallest stable max-rank loss over the entire run (with ties resolved sanely; see Line 38 of Algorithm 1

METHOD / SOLVER	Steps	Last Pareto [34] Mid ℓ_1-norm (↓)	Normalized ℓ_1-norm Minimizer Mid ℓ_1-norm (↓)	ℓ_1-rank Minimizer Mid ℓ_1-norm (↓)	Stable Max-Rank Minimizer Mid ℓ_1-norm (↓)	ℓ_1-norm Minimizer Mid ℓ_1-norm (↓)
Expert-Tuned	1	14.091	14.091	14.091	14.091	14.091
Proposed	3000	**11.089**	***10.995***	**11.014**	**10.995**	***10.995***
	6000	11.604	***10.973***	**10.988**	**10.988**	***10.973***
	12000	11.663	10.973	**10.988**	**10.988**	10.973
AGE-MOEA [38]	3000	11.546	*11.024*	11.427	***11.427***	*11.024*
	6000	11.523	10.997	*11.401*	11.443	10.997
	12000	11.502	10.957	11.374	11.625	10.957
RVEA [6]	3000	11.851	11.043	***11.399***	*11.437*	11.043
	6000	11.632	11.005	***11.372***	***11.399***	10.997
	12000	11.601	***10.954***	11.365	11.575	***10.954***
C-TAEA [28]	3000	11.622	**10.985**	*11.424*	11.461	**10.985**
	6000	***11.437***	*10.981*	11.401	*11.411*	*10.981*
	12000	11.586	**10.941**	***11.357***	*11.421*	**10.941**
Mosleh *et al.* [34]	3000	*11.497*	11.562	11.497	11.497	11.497
	6000	**11.387**	11.417	11.417	11.417	11.387
	12000	***11.417***	11.384	11.405	***11.405***	11.384
U-NSGA-III [46]	3000	13.422	11.026	11.449	11.512	11.026
	6000	*11.458*	**10.966**	11.405	11.633	**10.966**
	12000	**11.400**	10.964	11.365	11.539	10.964
NSGA-III [9, 10]	3000	***11.407***	11.063	11.532	11.574	11.063
	6000	11.736	11.015	11.404	11.561	11.015
	12000	*11.449*	10.957	*11.360*	11.610	10.957
R-NSGA-III [57]	3000	12.306	11.204	11.619	11.619	11.118
	6000	13.563	11.116	11.453	11.538	11.100
	12000	11.572	*10.957*	11.377	11.667	*10.957*
SMS-EMOA [5]	3000	13.658	11.062	11.456	11.750	11.062
	6000	11.635	11.010	11.414	11.652	11.010
	12000	11.486	11.010	11.377	11.595	10.988

Table 3. For each solver, value of the ℓ_1-norm of the loss vector $\boldsymbol{\mathcal{L}}$ of the selected champion for different champion selection methods after 3000, 6000 and 12,000 loss evaluations. For each of three optimization runs per solver, the median ℓ_1-norm is shown (see Table 4 for the min and max values). **Bold** denotes the best solver for a given selection method and total number of loss evaluations; ***bold italics*** shows the second best; *italics*, the third best. Solvers (expert-tuned excluded) are listed by increasing value of the median ℓ_1-norm of the loss vector of the stable max-rank loss champion at 3000 loss evaluations; this ordering is used in the figure that accompanies Table 1 in the main document.

in this supplementary document or Line 22 of Algorithm 1 in the main document). In this section, we compare these criteria with two others: the normalized ℓ_1-norm minimizer, and the ℓ_1-norm rank minimizer.

The (Weighted) Normalized ℓ_1-Norm. The *normalized ℓ_1-norm* of the loss vector is simply the weighted ℓ_1-norm of the loss vector where each component is divided by its standard deviation. In

METHOD / SOLVER	Steps	Last Pareto [34]		Normalized ℓ_1-norm Minimizer		ℓ_1-rank Minimizer		Stable Max-Rank Minimizer		ℓ_1-norm Minimizer	
		$\min \ell_1$ (↓)	$\max \ell_1$ (↓)	$\min \ell_1$ (↓)	$\max \ell_1$ (↓)	$\min \ell_1$ (↓)	$\max \ell_1$ (↓)	$\min \ell_1$ (↓)	$\max \ell_1$ (↓)	$\min \ell_1$ (↓)	$\max \ell_1$ (↓)
Expert Tuned	1	14.088	14.166	14.088	14.166	14.088	14.166	14.088	14.166	14.088	14.166
Proposed	3000	***11.051***	*11.965*	***10.970***	11.483	**10.970**	11.508	**10.970**	***11.508***	***10.970***	11.479
	6000	11.520	13.407	10.970	11.437	**10.970**	11.440	**10.970**	**11.445**	10.970	11.413
	12000	***11.154***	13.407	10.958	11.437	**10.958**	11.392	**10.958**	**11.392**	10.958	11.392
AGE-MOEA [38]	3000	*11.107*	***11.935***	**10.955**	*11.107*	*11.375*	***11.451***	*11.375*	**11.451**	**10.955**	***11.058***
	6000	*11.117*	11.839	*10.955*	*11.006*	11.375	11.427	11.416	11.707	*10.955*	*11.006*
	12000	11.385	11.730	10.955	11.012	11.365	11.379	11.575	11.638	10.955	11.006
RVEA [6]	3000	11.411	12.955	10.988	11.193	11.385	**11.421**	11.385	*11.526*	10.988	*11.098*
	6000	11.599	13.356	***10.954***	11.034	*11.365*	**11.383**	*11.365*	11.508	***10.954***	11.034
	12000	11.410	***11.602***	10.952	11.019	11.360	***11.371***	*11.390*	*11.602*	10.952	10.991
C-TAEA [28]	3000	11.456	**11.879**	*10.972*	11.144	11.405	11.461	11.424	11.563	*10.972*	11.144
	6000	11.422	**11.543**	**10.945**	***10.990***	11.393	***11.408***	11.402	***11.484***	**10.945**	***10.990***
	12000	11.408	11.665	**10.938**	**10.963**	11.353	**11.370**	11.400	***11.436***	**10.938**	**10.963**
Mosleh *et al.* [34]	3000	**10.989**	13.011	10.989	14.128	***11.005***	13.011	***11.005***	13.011	10.989	13.011
	6000	**11.004**	12.949	11.001	13.014	***10.979***	12.949	***10.977***	12.962	10.960	12.932
	12000	**10.953**	12.929	***10.940***	12.920	***10.972***	12.929	***10.972***	12.929	***10.940***	12.913
U-NSGA-III [46]	3000	11.524	14.884	11.000	**11.028**	11.405	*11.459*	11.501	11.572	11.000	**11.028**
	6000	11.417	***11.809***	10.964	**10.976**	11.366	*11.414*	11.572	11.637	10.964	**10.976**
	12000	11.379	*11.615*	10.959	***10.966***	11.361	*11.372*	11.526	11.627	10.959	***10.966***
NSGA-III [9, 10]	3000	11.113	13.824	10.991	***11.101***	11.409	11.541	11.451	11.663	10.991	11.101
	6000	11.368	13.352	10.983	11.035	11.372	11.423	11.508	*11.573*	10.983	11.035
	12000	*11.378*	**11.498**	10.953	*10.970*	11.358	11.378	11.458	11.615	10.953	*10.970*
R-NSGA-III [57]	3000	12.167	15.118	11.100	11.214	11.532	11.695	11.532	11.695	11.100	11.204
	6000	11.449	15.094	11.028	11.120	11.446	11.494	11.534	11.778	11.028	11.100
	12000	11.509	12.461	*10.951*	10.978	11.374	11.392	11.645	11.675	*10.951*	10.978
SMS-EMOA [5]	3000	11.291	14.036	11.056	11.107	11.435	11.557	11.495	11.880	11.056	11.107
	6000	***11.073***	*11.813*	10.994	11.080	11.405	11.490	11.395	11.723	10.994	11.080
	12000	11.485	11.639	10.988	11.034	*11.351*	11.378	11.588	11.626	10.974	11.034

Table 4. For each solver, value of the ℓ_1-norm of the champion's loss vector $\boldsymbol{\mathcal{L}}$ for different champion selection methods, after 3000, 6000 and 12,000 loss evaluations. For each of three optimization runs per solver, the lowest and highest ℓ_1-norms are shown. **Bold** denotes the best solver for a given selection method and total number of loss evaluations; ***bold italics*** shows the second best; *italics*, the third best. Solvers are listed in the same order as for Table 3.

particular, if the standard deviation of a loss component equals 0 (*e.g.*, when there is only one loss vector sample), this loss component is replaced by 0 since it obviously cannot guide champion selection. There is consequently no division by zero. Like the max-rank loss, stable or not, the normalized ℓ_1-norm is invariant with respect to rescalings of individual loss components. This champion selection criterion was inspired by [55, 56], where the corresponding total ordering is used to preprocess MOO data in order to speed up Pareto front computation. However, unlike this work's version, it is only approximately invari-

METHOD / SOLVER	Steps	Last Pareto Champion Selection Method [34] $\mathcal{L}_{\text{depth}}$ (↓) Low	Mid	High	$\mathcal{L}_{\text{int.}}$ (↓) High	Mid	Low	Stable Max-Rank Loss Selection (Proposed) $\mathcal{L}_{\text{depth}}$ (↓) Low	Mid	High	$\mathcal{L}_{\text{int.}}$ (↓) High	Mid	Low
Expert-Tuned	1	12.14	12.16	12.20	1.971	1.976	1.930	12.14	12.16	12.20	1.971	1.976	1.930
Proposed	3000	9.84	10.84	10.85	**2.129**	**0.243**	**0.216**	9.85	10.75	10.77	1.654	**0.243**	**0.216**
	6000	9.68	9.83	10.71	3.724	*1.692*	*0.899*	9.84	10.75	10.75	***1.608***	**0.243**	**0.216**
	12000	9.68	9.78	10.73	3.724	1.887	***0.424***	9.83	10.74	10.75	***1.565***	**0.243**	**0.216**
AGE-MOEA [38]	3000	9.19	9.84	10.88	*2.352*	2.096	***0.225***	**9.84**	**9.86**	**9.91**	*1.567*	1.542	1.537
	6000	9.17	9.68	10.90	2.351	2.163	**0.216**	*9.67*	9.83	9.86	2.042	1.613	1.558
	12000	9.13	*9.23*	10.68	2.376	2.155	*1.045*	***9.54***	9.76	9.76	2.038	1.878	1.869
RVEA [6]	3000	9.26	*9.51*	*9.91*	3.699	2.341	1.505	9.88	9.90	10.10	**1.537**	*1.505*	1.422
	6000	*8.93*	***9.10***	**9.25**	4.431	2.535	2.346	9.81	9.83	9.84	*1.702*	*1.555*	1.533
	12000	*9.07*	**9.08**	**9.17**	2.535	2.527	2.243	9.80	9.80	9.82	*1.806*	1.768	1.572
C-TAEA [28]	3000	9.17	9.80	***9.89***	2.708	1.819	1.566	9.88	9.92	10.15	***1.544***	1.537	*1.417*
	6000	9.15	9.32	*9.83*	2.270	2.120	1.714	9.89	9.93	10.02	**1.522**	*1.477*	*1.467*
	12000	9.78	9.92	10.29	**1.810**	***1.487***	1.373	9.90	9.94	9.94	**1.502**	*1.492*	*1.477*
Mosleh *et al.* [34]	3000	10.75	10.77	10.79	***2.220***	***0.727***	*0.243*	10.76	10.77	10.79	2.220	***0.727***	***0.243***
	6000	10.70	10.76	10.79	***2.164***	***0.688***	***0.243***	10.73	10.74	10.80	2.164	***0.677***	***0.243***
	12000	10.71	10.74	10.79	*2.142*	**0.677**	**0.243**	10.73	10.73	10.79	2.142	***0.677***	***0.243***
U-NSGA-III [46]	3000	**8.90**	***9.03***	**9.15**	5.986	4.393	2.371	***9.85***	*9.94*	***9.95***	1.721	1.564	1.560
	6000	9.29	9.78	9.87	*2.169*	2.032	1.546	***9.57***	***9.79***	***9.81***	2.063	1.845	1.762
	12000	***9.07***	***9.11***	***9.58***	2.335	2.267	2.038	**9.49**	**9.49**	9.74	2.045	2.041	1.886
NSGA-III [9, 10]	3000	***8.93***	9.85	10.86	4.892	*1.561*	0.252	9.93	9.96	10.00	1.707	1.576	1.521
	6000	***8.92***	*9.16*	***9.54***	4.429	2.205	2.198	9.81	*9.82*	*9.83*	1.762	1.740	1.676
	12000	9.34	9.86	10.02	1.474	9.73	***2.114***	*1.52*	9.77	9.79	1.884	1.837	1.671
R-NSGA-III [57]	3000	9.05	9.65	10.80	6.065	2.652	1.372	10.00	10.06	10.16	1.564	1.531	1.528
	6000	**8.91**	**8.94**	9.92	6.183	4.624	1.528	9.86	9.87	9.90	1.914	1.671	1.633
	12000	**9.02**	9.76	9.81	3.444	1.815	1.697	9.61	*9.72*	*9.73*	2.039	1.944	1.941
SMS-EMOA [5]	3000	*8.93*	**8.96**	10.87	5.102	4.698	0.412	*9.85*	***9.87***	*9.98*	2.029	1.767	1.629
	6000	9.57	10.70	10.78	**2.070**	**0.294**	1.110	**9.22**	**9.66**	**9.68**	2.177	2.043	1.989
	12000	9.11	9.41	*9.75*	2.371	2.079	1.890	*9.55*	***9.56***	**9.68**	2.040	2.033	1.948

Table 5. For each solver, value of the point cloud's average RMSE depth loss $\mathcal{L}_{\text{depth}}$ and RMSE intensity loss $\mathcal{L}_{\text{int.}}$ for the last Pareto point up to the given number of loss evaluations (left two columns), and stable max-rank loss champion (right) after 3000, 6000 and 12,000 loss evaluations. For each of three optimization runs per solver, as well as for the expert-tuned configuration, the lowest, median and highest values of the loss are shown. **Bold** shows the best solver for a given champion selection method and total number of loss evaluations; ***bold italics*** shows the second best; *italics*, the third best. Solvers are listed in the same order as for Table 3.

ant with respect to individual loss component rescalings since division by a vanishing standard deviation is avoided by adding a small positive number (an "epsilon").

The (Stable Weighted) ℓ_1-Rank. The ℓ_1-*rank* of the loss vector is obtained by aggregating the (stable) ranks of each component with the (weighted) ℓ_1-norm instead of the (weighted) ℓ_∞-norm, that is, by replacing "$\max$" by "Σ" in Eq. (13) of the main document.

We also compare with the minimizer of the ℓ_1-norm of the loss vector, our evaluation metric (see

Method / Solver	Steps	Last Pareto [34] $\Delta\mathcal{L}_{\text{depth}}$ (↓)	Last Pareto [34] $\Delta\mathcal{L}_{\text{int}}$ (↓)	Normalized ℓ_1-norm Minimizer $\Delta\mathcal{L}_{\text{depth}}$ (↓)	Normalized ℓ_1-norm Minimizer $\Delta\mathcal{L}_{\text{int}}$ (↓)	ℓ_1-rank Minimizer $\Delta\mathcal{L}_{\text{depth}}$ (↓)	ℓ_1-rank Minimizer $\Delta\mathcal{L}_{\text{int}}$ (↓)	Stable Max-Rank Minimizer $\Delta\mathcal{L}_{\text{depth}}$ (↓)	Stable Max-Rank Minimizer $\Delta\mathcal{L}_{\text{int}}$ (↓)	ℓ_1-norm Minimizer $\Delta\mathcal{L}_{\text{depth}}$ (↓)	ℓ_1-norm Minimizer $\Delta\mathcal{L}_{\text{int}}$ (↓)
Proposed	3000	1.010	***1.914***	0.827	1.341	0.917	1.438	0.900	1.438	0.858	1.368
	6000	1.021	2.825	0.866	1.333	0.922	1.392	0.916	1.392	0.895	1.338
	12000	1.046	3.300	0.855	1.333	0.918	1.349	0.918	1.349	0.916	1.349
AGE-MOEA [38]	3000	1.687	2.127	0.142	*0.009*	0.071	**0.032**	*0.071*	**0.032**	0.060	0.078
	6000	1.730	2.136	***0.042***	0.028	0.035	0.026	0.192	0.484	*0.042*	0.028
	12000	1.558	1.330	0.064	0.009	0.032	0.018	0.224	0.169	0.049	0.009
RVEA [6]	3000	*0.650*	2.194	0.169	0.036	*0.037*	0.067	0.224	*0.115*	***0.046***	0.091
	6000	***0.328***	2.085	0.051	0.036	*0.026*	**0.013**	*0.038*	*0.170*	0.044	0.036
	12000	***0.101***	**0.292**	0.057	0.009	*0.016*	0.028	**0.022**	0.234	**0.003**	0.036
C-TAEA [28]	3000	0.720	**1.143**	0.186	0.027	0.070	***0.033***	0.265	0.127	0.186	*0.027*
	6000	0.677	**0.556**	0.045	0.027	***0.014***	0.024	0.128	**0.055**	0.045	0.027
	12000	0.515	*0.437*	**0.005**	0.027	0.025	*0.012*	***0.046***	**0.025**	***0.005***	0.027
Mosleh *et al.* [34]	3000	**0.045**	*1.976*	1.419	1.720	***0.029***	1.976	**0.029**	1.976	0.045	1.976
	6000	**0.086**	1.921	0.143	1.916	0.049	1.921	0.064	1.921	0.069	1.921
	12000	**0.078**	1.899	0.081	1.899	0.060	1.899	0.060	1.899	0.052	1.921
U-NSGA-III [46]	3000	***0.254***	3.615	**0.027**	***0.009***	**0.016**	0.047	0.097	0.162	**0.027**	***0.009***
	6000	*0.582*	***0.623***	**0.012**	$\sim$ **0.0**	**0.013**	0.041	0.240	0.301	**0.012**	$\sim$ **0.0**
	12000	*0.511*	***0.297***	***0.007***	$\sim$ **0.0**	***0.009***	**0.002**	0.257	0.159	*0.007*	$\sim$ **0.0**
NSGA-III [9, 10]	3000	1.929	4.640	*0.074*	0.036	0.167	0.038	***0.070***	0.184	0.074	0.036
	6000	0.616	2.232	*0.042*	***0.009***	0.039	***0.014***	**0.021**	***0.086***	0.042	***0.009***
	12000	0.688	0.640	*0.016*	$\sim$ *0.0*	**0.002**	0.018	*0.057*	0.214	0.016	$\sim$ *0.0*
R-NSGA-III [57]	3000	1.742	4.693	0.167	0.054	0.160	*0.036*	0.160	***0.036***	0.166	0.080
	6000	1.010	4.655	0.102	0.027	0.041	*0.015*	***0.037***	0.281	***0.028***	0.044
	12000	0.795	1.747	0.027	$\sim$ ***0.0***	0.017	***0.005***	0.128	*0.098*	0.027	$\sim$ ***0.0***
SMS-EMOA [5]	3000	1.940	4.685	***0.053***	**0.009**	0.112	0.041	0.131	0.400	*0.053*	**0.009**
	6000	1.214	*1.776*	0.094	*0.009*	0.080	0.015	0.461	0.188	0.094	*0.009*
	12000	0.634	0.481	0.055	0.009	0.019	0.016	0.131	***0.092***	0.087	0.027

Table 6. For each solver, difference between the maximal and minimal average RMSE depth loss $\Delta\mathcal{L}_{\text{depth}}$ and RMSE intensity loss $\Delta\mathcal{L}_{\text{intensity}}$ across 3 different optimization runs for different champion selection methods after 3000, 6000 and 12,000 loss evaluations. **Bold** highlights the lowest difference between average losses for a given champion selection method and total number of loss evaluations, ***bold italics*** depicts the second lowest, while *italics* denotes the third lowest. Solvers order is the same as for Table 3.

Section 1.4). Clearly, the normalized ℓ_1-norm is a good match to the ℓ_1-norm when loss components are well-balanced, as is the case here. Additional existing champion selection methods, like the *pseudo-weight* [8], were not tested.

In Table 3, we show, for each solver, the median, over the three runs, of the ℓ_1-norm of champions chosen with various methods; we also show, in the last column, the minimizer of the ℓ_1-norm, our evaluation metric. For short runs (3000 loss evaluations), the proposed solver gets either the best or second best result, no matter the champion selection method. The proposed solver also ranks well with

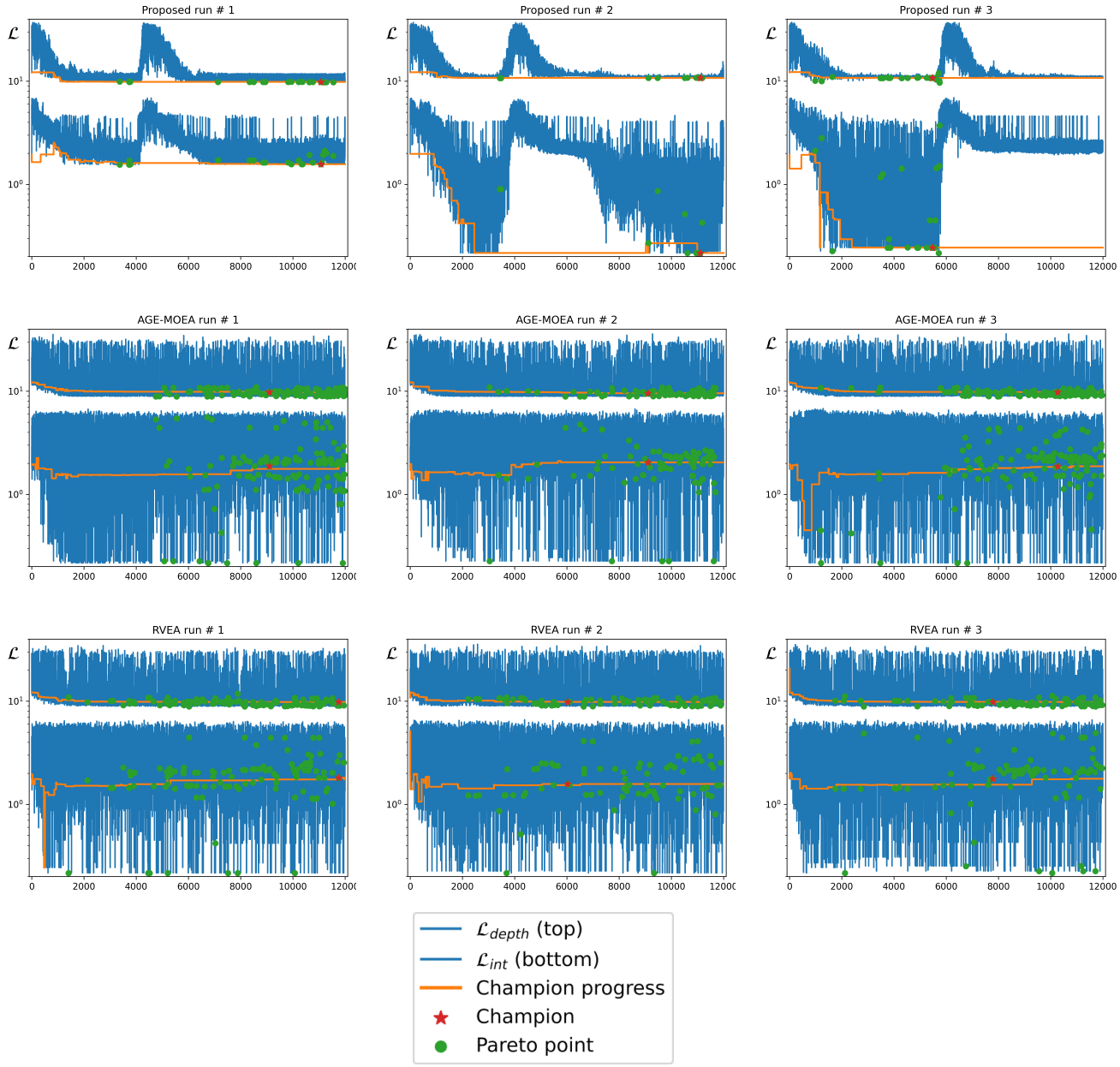

Figure 3. MOO convergence plots where the optimized objectives were the $\text{RMSE}_{\text{depth}}$ and $\text{RMSE}_{\text{intensities}}$ for the proposed, AGE-MOEA [38] and RVEA [6] solvers. The blue curves correspond to the loss values $\mathcal{L}_{\text{depth}}$ (top) and $\mathcal{L}_{\text{intensities}}$ (bottom) while the orange curve tracks the loss of the max-rank champion at each step. The red star denotes where the final champion was found during the run and the green circles represent the Pareto points relative to the run's data (they may not be actual Pareto points, only approximations). The proposed method converges to the same local minimum as several of the other methods in the first run, and converges near an ℓ_1-norm minimizer in the other two.

longer optimization runs. Table 4, instead of the median of three runs, shows the min and the max. Again, the proposed solver performs well when the run is short or when the champion is selected using

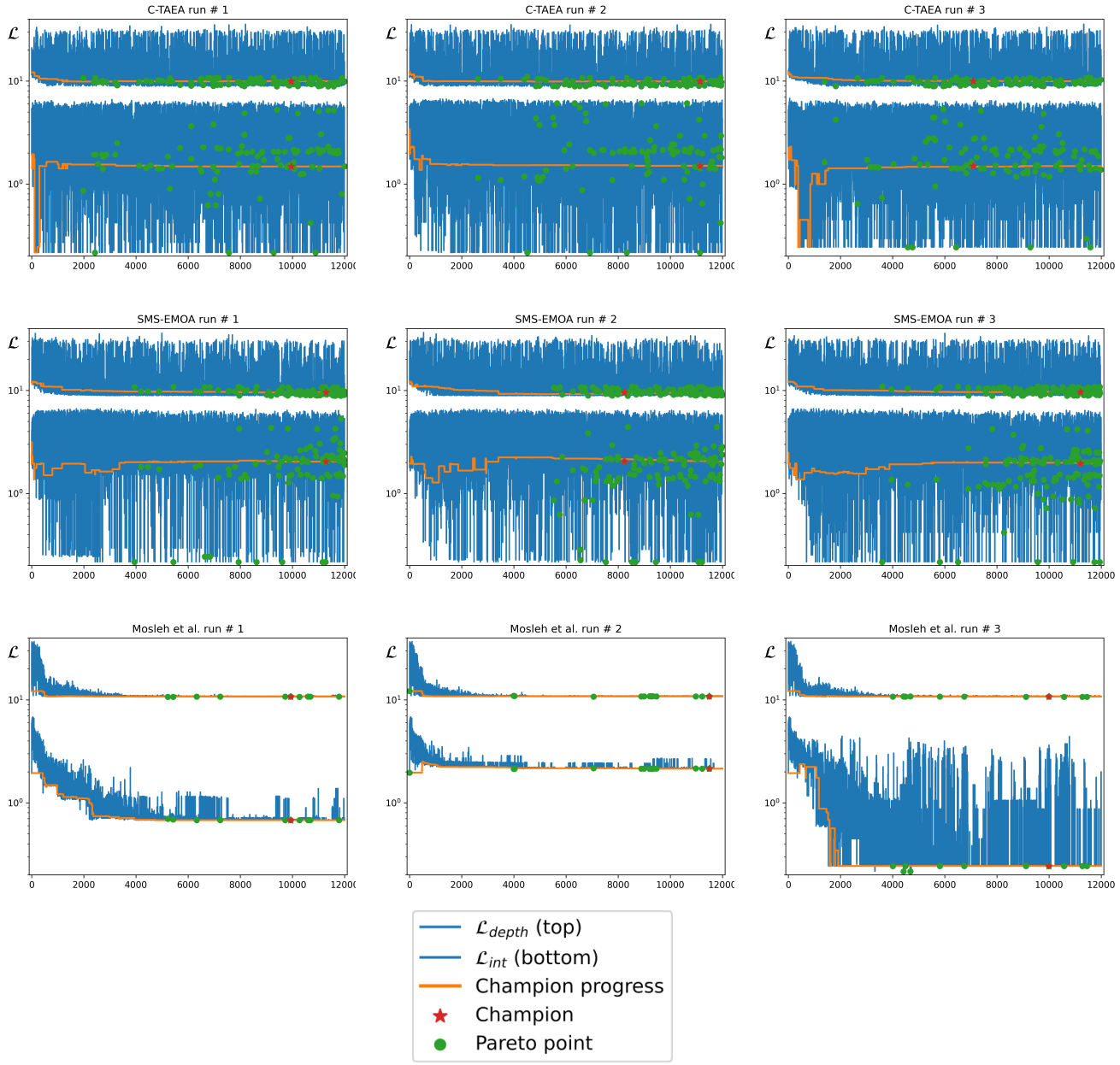

Figure 4. MOO convergence plots where the optimized objectives were the $\text{RMSE}_{\text{depth}}$ and $\text{RMSE}_{\text{intensities}}$ for the C-TAEA [28], Mosleh *et al.* [34] and SMS-EMOA [5] solvers. The blue curves correspond to the loss values $\mathcal{L}_{\text{depth}}$ (top) and $\mathcal{L}_{\text{intensities}}$ (bottom) while the orange curve tracks the loss of the max-rank champion at each step. The red star denotes where the final champion was found during the run and the green circles represent the Pareto points relative to the run's data (they may not be actual Pareto points, only approximations).

the stable max-rank loss. (This is unsurprising given that the stable max-rank loss drives optimization with the proposed method.) Other solvers also get comparatively good results; in part, this is because, as seen in Figure 3, one of the three runs performed with the proposed solver converges to the same attractor as most of the other runs (see the figure that accompanies Figure 1 in the main document).

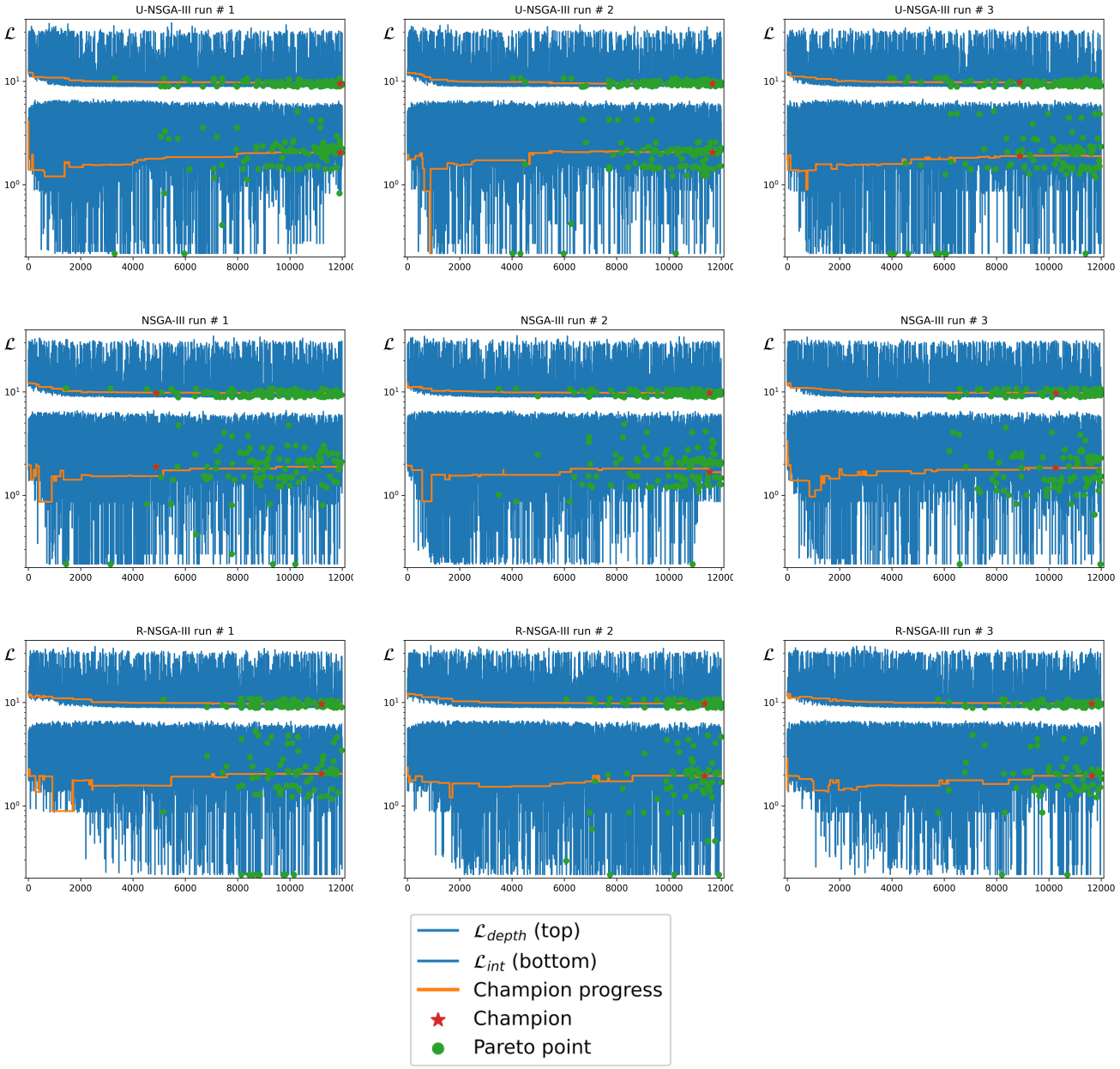

Figure 5. MOO convergence plots where the optimized objectives were the $\text{RMSE}_{\text{depth}}$ and $\text{RMSE}_{\text{intensities}}$ for the U-NSGA-III [46], NSGA-III [9, 10] and R-NSGA-III [57] solvers. The blue curves correspond to the loss values $\mathcal{L}_{\text{depth}}$ (top) and $\mathcal{L}_{\text{intensities}}$ (bottom) while the orange curve tracks the loss of the max-rank champion at each step. The red star denotes where the final champion was found during the run and the green circles represent the the Pareto points relative to the run's data (they may not be actual Pareto points, only approximations).

Tables 3 and 4 also confirm that all three tested champion selection methods improve on *last Pareto of the run.* (As explained in Section 1.4, the ℓ_1-norm of the loss, our evaluation metric, is out of the running as a champion selection method because it is not invariant with respect to individual loss component rescalings.)

Leaving the ℓ_1-norm aside, Table 5 shows all three of the lowest, median and highest values of each champion's loss components after 3000, 6000 and 12,000 loss evaluations for each solver. In this table, we only compare two champion selection methods: the *last Pareto of the run* selection criterion, and the proposed *minimizer of the (stable) max-rank loss*. Looking at loss component values, we note that within the Pareto front, the two losses are in opposition, as is always the case when there are only two losses: by definition, decreasing one loss must increase the other. Thus, for solvers that converge close to the actual Pareto front, a low value of one loss means a high value for the other. Consequently, values in the middle of the range are not necessarily inferior even though one would ideally want both losses to be as low as possible. This explains why increasing the number of iterations does not always decrease the value of both losses: Pareto points, from which the champions are selected, are compromises. In any case, the proposed solver again fares well, especially when the total number of loss evaluation is low, and the proposed max-rank loss champion selection method is again seen to generally improve on *last Pareto point of the run*; see, in particular, the values of $\mathcal{L}_{\text{int.}}$.

Table 6 shows Δ, the difference between the high and low values of each loss component, for the champions obtained with each selection method and solver. Δ corresponds to the vertical distance between the farthest champions in a row of convergence plots of Figures 3, 4 and 5. A smaller Δ suggests better stability. Here, the proposed solver fares poorly. The reason for this is that, as seen in the first row of Fig. 3, the proposed solver sometimes converges to the local minimum of the Pareto front where the champions of many of the other methods converge, instead of the ℓ_1-norm minimizer: In the figure that accompanies Table 1 in the main document, most methods' champions are clustered around a central point of the Pareto front where it juts toward the origin. This suggests that solvers that "sweep" more than they "converge", like the hypervolume-based methods tested in this work, may have better stability than methods that aggressively seek minima like the proposed method and the method of Mosleh *et al*.

Table 6 also shows that all three of the above proposed champion selection methods, namely normalized ℓ_1-norm, ℓ_1-rank and max-rank loss, improve on *last Pareto of the run*. Which of the three is the best overall is unclear. In addition, stability does not seem well correlated with result quality.

1.7. Training Details for 3D Detector

To generate training data, we use the proposed LiDAR sensing model within the CARLA simulation environment. First, we spawn the ego-vehicle, 60 other vehicles and 50 pedestrians randomly in the world. To animate the world, we use the traffic manager and walker artificial intelligence of the CARLA engine so that the actors move towards random destinations within the map. At every step of the simulation, we convert the bounding boxes of all available actors to the KITTI format. Since CARLA returns bounding boxes for all actors within the map, we filter out bounding boxes without a single point. Furthermore, pedestrian bounding boxes, as extracted directly from CARLA, do not encapsulate the full pedestrian during, *e.g.*, walking. We thus refine these by iterating through the skeleton and appropriately enlarging the bounding box. Then, we emulate a sequence of 500 simulation steps with step size 0.1s. We collect 10 sequences for five of the available CARLA worlds. For every world, we hold back one sequence as validation set and use the remaining ones for training. We thus select 5900 frames in total for training. We ensure that every frame has valid bounding boxes with at least a single point and use only every fourth frame in a sequence. For validation, we select 1000 frames using the same procedure. To ensure that the object detector performs reasonably for different hyperparameter settings $\boldsymbol{\Theta}$, we use 8 different parameterizations as defined in Table 7. Hyperparameters are selected manually ensuring that

they generate point clouds with different quantities of clutter points. Although hyperparameters could be made to depend on the channel m, we use the same hyperparameter value over all channels.

Θ	SELECTED VALUES							
P_0	200	500	200	1000	500	750	750	100
V	0.00	0.02	0.02	0.05	0.05	0.02	0.05	0.05
τ	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0

Table 7. Hyperparameters $\boldsymbol{\Theta}$ varied for generating training data. Hyperparameters are defined as globally constant for each channel m.

We chose the PV-RCNN detector [48] as a representative 3D detector and train on full range 360° point clouds from scratch. For training, the standard augmentation methods—flipping, rotation and copy-paste ground truth augmentation—are used. We adopt the training configuration of [48] but decrease the learning rate to 0.001 to ensure better convergence. Note that the detector is trained only on cars and pedestrians. Trucks and cyclists are separate classes; the detector is not trained on them.

1.8. Additional Object Detection Results

We provide additional qualitative object detection results in Fig. 6 and Fig. 7. We compare the expert-tuned and optimized point clouds with the ground truth. As described in the main paper, optimization results in the removal of clutter and produces accurate point clouds that closely resemble the ground truth. This comes at the cost of missing ground points at farther distances. This, however, does not affect the detection results for farther away objects. Pedestrian detection especially improves. Pedestrians are often missed within the expert-tuned point cloud because of clutter. For cars, fewer false positives occur and the alignment of the bounding boxes qualitatively improves over the expert-tuned point cloud, resulting in higher IoU and higher average precision (AP).

1.9. Off-the-Shelf LiDAR Experiment

Unlike what is the case with the above LiDAR simulations for which the ground truth depth map is readily accessible, we do not have fine control over the scanning parameters of the specific LiDAR system used for the real LiDAR experiments and, consequently, cannot easily acquire a ground truth for the off-the-shelf LiDAR experiments. If the vendor provided full access to the sensing and DSP pipelines, an accurate ground truth could be acquired similarly to Gruber *et al*. [16] by dense (and slow) scanning of a static scene.

As only fixed scanning patterns are available for the off-the-shelf LiDAR hardware employed in our investigation, a direct pointwise metric is challenging to use; instead, we work with 3D histograms of sampled point clouds. These histograms are defined with 3D bins expressed in spherical coordinates with each axis sampled uniformly. The histogram bin volume depends on the distance and angles relative to the origin; farther bin volumes are scaled by distance squared. We sample bins with 20 cm range resolution, 0.1° resolution for the vertical angles and 0.2° resolution for the horizontal angles.

We denote by h^{GT} the ground truth 3D histogram obtained by averaging 100 point cloud frames generated with uniform scanning pattern angular resolution. The loss function used for the optimization

Figure 6. Qualitative results with object detection as optimization objective. For visualization purposes, only the camera field-of-view is shown.

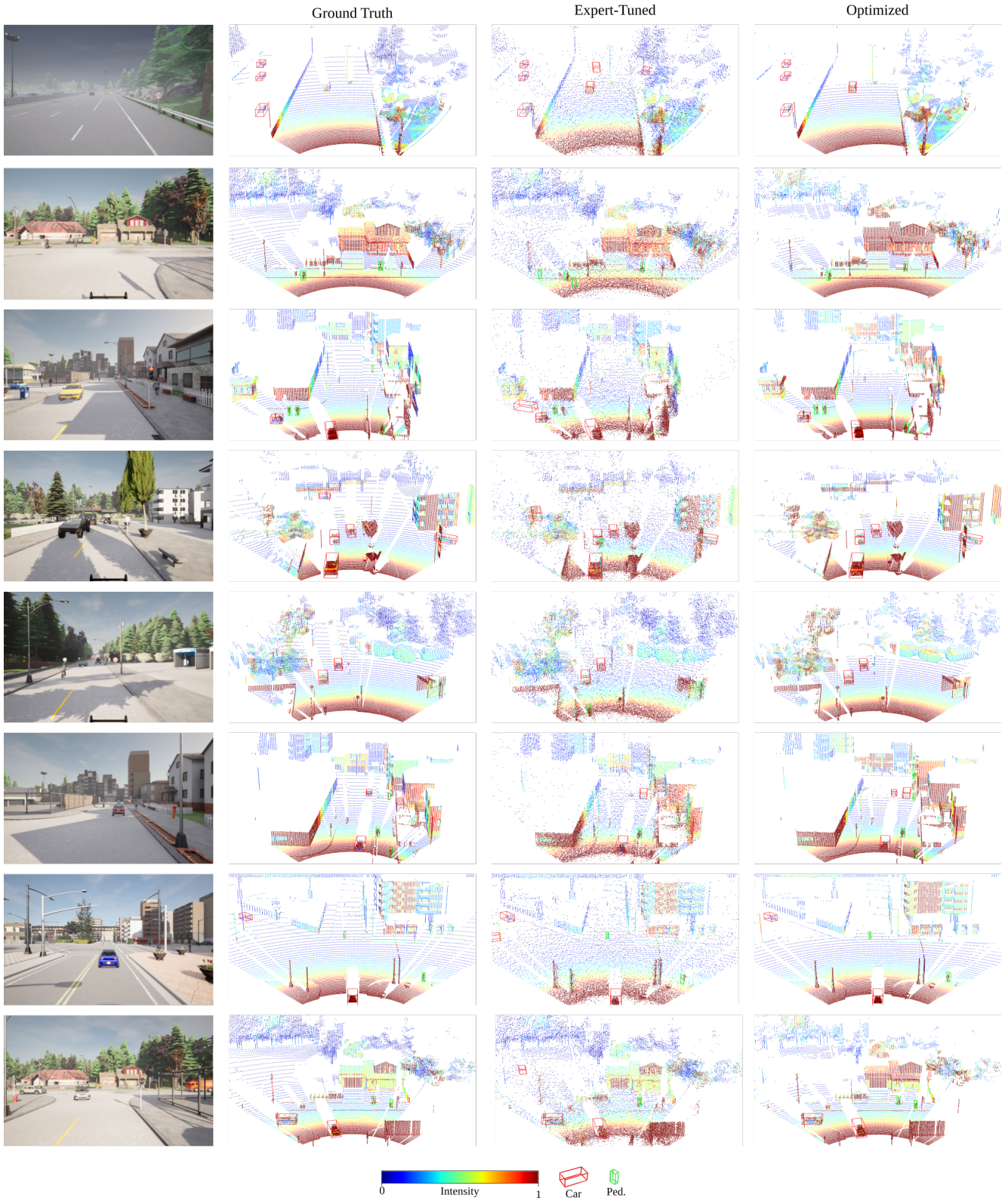

Figure 7. Qualitative results with object detection as optimization objective. For visualization purposes, only the camera field-of-view is shown.

METRIC	Optimized	Expert
$\lvert\mathcal{L}_{\text{3D hist.}}\rvert/10^5$ ($\uparrow$)	**1.59**	1.32
$\text{RMSE}_{\text{depth}}$ ($\downarrow$)	**2.98** cm	10.1 cm

Table 8. Off-the-shelf LiDAR experiment loss and $\text{RMSE}_{\text{depth}}$ evaluation for expert and optimized parameters. The best value for each metric is in **bold**.

experiment is

$$\mathcal{L}_{\text{3D hist.}}(\boldsymbol{\Theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{n=1}^{N_{\text{bins}}}(h^{\text{GT}} \otimes h^i(\boldsymbol{\Theta}))_n, \tag{3}$$

where $N = 10$ is the number of frames taken to average the histogram multiplication, $h^i(\boldsymbol{\Theta})$ is the i-th point cloud histogram taken for the hyperparameter configuration $\boldsymbol{\Theta}$, $\otimes$ denotes bin-wise multiplication and the inner sum runs over all histogram bins. This loss function drives the optimizer to accumulate as many 3D points as possible where the ground truth is denser, as seen in Fig. 6 of the main paper where the histogram binwise multiplication is shown prior to summing. The optimized configuration yields denser point clouds where the ground truth is dense while the expert-tuned point cloud had a different ROI where the ground truth was not dense, resulting in a high loss. The fitness function mentioned in the main article is the negative of the loss function of Eq. 3. A feature of this loss function is that higher bin counts than the ground truth histogram are not penalized; this needs to be considered when computing a quantitative depth error metric for the histograms. As such, we choose the following metric

$$\text{RMSE}_{\text{depth}} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{N_{\text{bins}}}\sum_{n=1}^{N_{\text{bins}}}\mathcal{D}\left(h_n^{\text{GT}}, h_n^i\right)^2 r_n^2\sqrt{h_n^i}}, \tag{4}$$

where r_n is the depth of the edge closest to the LiDAR unit of the n-th bin and

$$\mathcal{D}(h_n^{\text{GT}}, h_n^i) = \begin{cases} 0 & \text{if } h_n^{\text{GT}} = 0 \text{ or } h_n^i = 0, \\ \max(0, \min(h_n^i, (h_n^{\text{GT}} - h_n^i))) & \text{otherwise.} \end{cases} \tag{5}$$

Here, $\mathcal{D}$ acts as a clipped difference in which point cloud histograms with higher bin counts than the ground truth are not penalized. Furthermore, we discard any bin where one or the other histogram registered no points in order to avoid unfair comparisons. Finally, each bin is weighted by the square root of the number of points registered, which gives more weight to higher sampling. Table 8 compares this metric to the loss function for both expert tuning and optimized parameters.

2. Additional Details on LiDAR Simulation and DSP

We model the radiance Λ_o from a given object located at $\boldsymbol{R}_i$ in 3D space projected onto a sensor at $\boldsymbol{S}$ with the rendering equation [24]

$$\Lambda_o(\mathbf{v}, \lambda) = \Lambda_e(\mathbf{v}, \lambda) + \int_{\mathbb{S}^+_{-\mathbf{n}}} \Lambda_i(\mathbf{l}, \lambda)\text{BRDF}(\mathbf{v}, \mathbf{l}, \mathbf{n}, \lambda)(\mathbf{n} \cdot \mathbf{l})d\mathbf{l}, \tag{6}$$

where Λ_e is the light intensity emitted at $\mathbf{R}_i$ coming onto $\mathbf{S}$, $\mathbf{v} = (\mathbf{S} - \mathbf{R}_i) / \|\mathbf{S} - \mathbf{R}_i\|$ is a unit vector pointing from $\mathbf{R}_i$ to the sensor, $\mathbf{n}$ is the surface normal at $\mathbf{R}_i$, $\mathbf{l}$ is a unit vector pointing from $\mathbf{R}_i$ to a light source, $\Lambda_i(\mathbf{l}, \lambda)$ is the light intensity coming from the direction $\mathbf{l}$ and reflected upon arrival, $\text{BRDF}(\mathbf{v}, \mathbf{l}, \mathbf{n}, \lambda)$ is the Bidirectional Reflectance Distribution Function (BRDF) describing the material reflectance at $\mathbf{R}_i$, and λ denotes the light wavelength. The integral is carried over $\mathbb{S}^+_{-\mathbf{n}}$ which is defined as the unit hemisphere whose base normal vector is equal to $-\mathbf{n}$. Below, we derive the special case of Eq. (6) for a single emitted LiDAR pulse.

2.1. LiDAR BRDF

LiDAR sensors use narrow band infrared lasers and narrow bandpass filters on the detector side. Therefore, λ can be treated as confined to a narrow part of the near-infrared spectrum (NIR). For simplicity, we assume λ to be constant and drop it from the equations. We split $\Lambda_i = \Lambda_{\text{LiDAR}} + \Lambda_{\text{ambient}}$ into two terms, namely the light coming from the LiDAR unit itself (Λ_{LiDAR}), as a LiDAR is its own light source, and light from other sources (Λ_{ambient}). We thus rewrite Eq. (6) as

$$\Lambda_o(\mathbf{v}) = \int_{\mathbb{S}^+_{-\mathbf{n}}} \Lambda_{\text{LiDAR}}(\mathbf{l}) \text{BRDF}(\mathbf{v}, \mathbf{l}, \mathbf{n})(\mathbf{n} \cdot \mathbf{l}) d\mathbf{l} + a(t), \tag{7}$$

with

$$a(t) = \Lambda_e(\mathbf{v}) + \int_{\mathbb{S}^+_{-\mathbf{n}}} \Lambda_{\text{Ambient}}(\mathbf{l}) \text{BRDF}(\mathbf{v}, \mathbf{l}, \mathbf{n})(\mathbf{n} \cdot \mathbf{l}) d\mathbf{l}. \tag{8}$$

In $a(t)$ we bundle all the light not emitted by the LiDAR itself. $a(t)$ is defined as the ambient illumination component and is described in more depth in Section 2.3. For Λ_{LiDAR}, all light is coming from the direction $\mathbf{v}$. Hence, we model the incoming radiation approximately as a beam with Dirac distribution $\Lambda_{\text{LiDAR}}(\mathbf{l}) = \Lambda_0 \delta(\mathbf{l} - \mathbf{v})$ where Λ_0 is the light intensity transmitted towards the object by the LiDAR. This Dirac distribution only leaves us with the retroreflected component of the BRDF,

$$\Lambda_o(\mathbf{v}) = \Lambda_0(\mathbf{l}) \text{BRDF}(\mathbf{v}, \mathbf{v}, \mathbf{n})(\mathbf{n} \cdot \mathbf{v}) + a(t). \tag{9}$$

After rearranging the equation we can then define the object's reflectivity ρ_i from the scene reflectivity as follows,

$$\rho_i := \frac{\Lambda_o(\mathbf{v}) - a(t)}{\Lambda_0} = \text{BRDF}(\mathbf{v}, \mathbf{v}, \mathbf{n})(\mathbf{n} \cdot \mathbf{v}). \tag{10}$$

We model this ρ_i with the Cook-Torrance model [7]: a microfacet model which describes a material surface as a collection of smaller smooth facets oriented in random directions. It acts as a statistical model averaging the combined effects of each microfacets. The model splits the BRDF into a linear combination of two terms: a specular and a diffusive part which can be written as

$$\text{BRDF}(\mathbf{v}, \mathbf{l}, \mathbf{n}) = s\text{BRDF}_s(\mathbf{v}, \mathbf{l}, \mathbf{n}) + d\text{BRDF}_d(\mathbf{v}, \mathbf{l}, \mathbf{n}), \tag{11}$$

where s and BRDF_s (resp. d and BRDF_d) are the specular coefficient and the specular part of the BRDF (resp. the diffuse coefficient and the diffuse part of the BRDF). The s and d coefficients are discussed in Section 2.2. For our simulation, we used a Lambertian model for the diffusive part, that is

$$\text{BRDF}_d(\mathbf{v}, \mathbf{l}, \mathbf{n}) = 1. \tag{12}$$

The specular part of the Cook-Torrance BRDF is

$$\text{BRDF}_s(\mathbf{v},\mathbf{l},\mathbf{n}) = \frac{\mathcal{F}(\mathbf{v},\mathbf{h})D(\mathbf{h},\mathbf{n})G(\mathbf{v},\mathbf{l},\mathbf{n})}{4\,(\mathbf{n}\cdot\mathbf{l})\,(\mathbf{n}\cdot\mathbf{v})}, \tag{13}$$

where $\mathbf{h} = (\mathbf{v}+\mathbf{l})/\|\mathbf{v}+\mathbf{l}\|$ is a halfway unit vector between $\mathbf{v}$ and $\mathbf{l}$, $\mathcal{F}$ is a Fresnel term describing how the light is reflected on each smooth microfacet, G is a geometrical factor averaging the effects of shadowing and masking by microfacets onto each other and D is a statistical distribution of the number of facets facing in the direction of $\mathbf{h}$. For $\mathcal{F}$, we adopted the Schlick's approximation [45], that is

$$\mathcal{F}_{\text{Schlick}}(\mathbf{v},\mathbf{h}) = \mathcal{F}_0 + (1-\mathcal{F}_0)(1-\mathbf{v}\cdot\mathbf{h})^5, \tag{14}$$

where $\mathcal{F}_0$ is the specular reflectance for wavelength λ. In the case of retroreflection, $\mathbf{v}=\mathbf{h} \;\rightarrow\; \mathbf{v}\cdot\mathbf{h}=1$ and therefore $\mathcal{F}(\mathbf{v},\mathbf{h}) = \mathcal{F}_0$ is reduced to a constant. For G, we adopt the same hybrid approach used by the CARLA engine to render the images [11], where we also adapt the Schlick's approximation of the G function derived in the GGX model by Walter *et al.* [45, 59]. This approach introduces a roughness parameter $\alpha \in [0,1]$ where $\alpha = 0$ corresponds to completely smooth surfaces which causes mirror-like reflections and where $\alpha \to 1$ defines very rough surfaces with almost no specular light propagation. As described by Smith *et al.* [50], Schlick *et al.* [45] and Walter *et al.* [59], this function can be approximated by a separable product of visibility functions $G(\mathbf{v},\mathbf{l},\mathbf{n}) = \mathcal{G}(\mathbf{v},\mathbf{n})\mathcal{G}(\mathbf{l},\mathbf{n})$ where $\mathcal{G}(\mathbf{v},\mathbf{n})$ (resp. $\mathcal{G}(\mathbf{l},\mathbf{n})$) describes the amount of reflected (resp. incident) non-obstructed light and

$$\mathcal{G}(\mathbf{x},\mathbf{n}) = \frac{\mathbf{x}\cdot\mathbf{n}}{(\mathbf{x}\cdot\mathbf{n})(1-k)+k}, \tag{15}$$

where $k = (\alpha+1)^2/8$. In the case of retroreflection where $\mathbf{l}=\mathbf{v}$, G reduces to

$$G(\mathbf{v},\mathbf{v},\mathbf{n}) = \frac{\cos^2\theta}{[\cos\theta(1-k)+k]^2}, \tag{16}$$

where we introduce θ as the angle between the surface normal $\mathbf{n}$ and the incident laser ray $\mathbf{l}$. Then, for the microfacet normal vector distribution function D, we adopted the Trowbridge-Reitz microfacet model [52], that is

$$D(\mathbf{h},\mathbf{n}) = \frac{\alpha^4}{((\mathbf{h}\cdot\mathbf{n})^2(\alpha^4-1)+1)^2}, \tag{17}$$

where we reuse the roughness parameter α as described above. In the retroreflective case where $\mathbf{h}=\mathbf{v}$, we can reduce equation (17) to

$$D(\mathbf{h},\mathbf{n}) = \frac{\alpha^4}{(\cos^2\theta(\alpha^4-1)+1)^2}. \tag{18}$$

Note that in the limit of $\alpha \to 1$, this function is uniform for all incident angles. Finally, by combining equations (14), (16) and (18) into (13), then combining Eq. (12) and (13) following Eq.(11) with the definition of Eq. (10) while using the fact that $\mathbf{v}\cdot\mathbf{n} = \cos\theta$, we obtain Eq. (7) of the main paper written as

$$\rho_i = \frac{\alpha^4 s\cos\theta}{4[\cos^2\theta(\alpha^4-1)+1]^2[\cos\theta(1-k)+k]^2} + d\cos\theta, \tag{19}$$

nwhere the d, s and α parameters are set as described in Section 2.2.

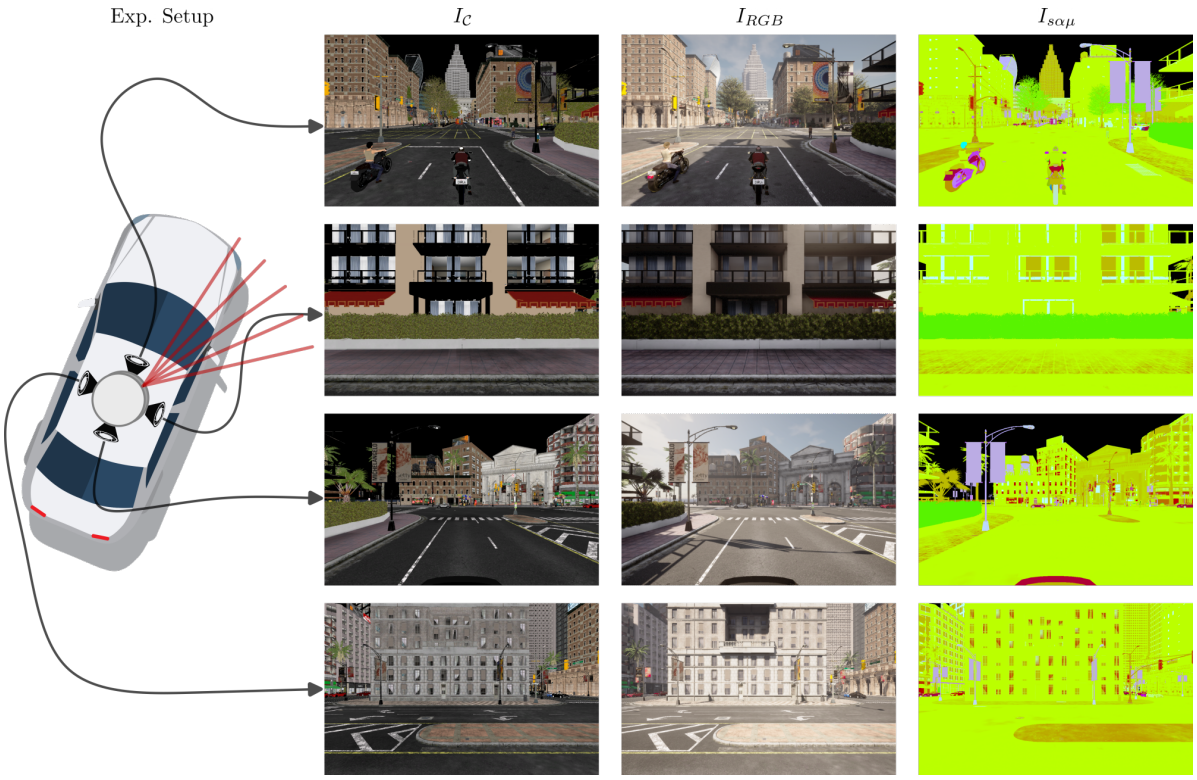

Figure 8. Qualitative illustration of the rendered images $I_{\mathcal{C}}$, $I_{s\alpha\mu}$ and I_{RGB} done with our proposed shader in the CARLA simulation pipeline.

2.2. Extracting BRDF Coefficients α, s and d from CARLA

In CARLA, every non-transparent material is described by its diffuse color $\mathcal{C}$, its specular component $s \in [0, 1]$, its roughness $\alpha \in [0, 1]$, and its metallic parameter $\mu \in [0, 1]$ where $\mu = 1$ defines a metal and $\mu = 0$, a dielectric. We extract these necessary values through a shader. The shader is applied to the LiDAR full 360° field of view and returns encoded images defined as $I_{sd\alpha}$. The shaders are spawned as cameras and the position setup is depicted on the left of Figure 8 where each camera span a 90° field of view and are rotated 90° with respect to each other. In particular, different shaders allow us to obtain information like a depth map, semantic map or the optical flow from CARLA [11]. The s, d and α parameters for each point $\mathbf{R}_i$ in the point cloud is then extracted from $I_{sd\alpha}$ following the procedure detailed in Section 2.4.

To generate $I_{sd\alpha}$, our shader firstly obtains two three-channel-wide images $I_{\mathcal{C}}$ and $I_{s\alpha\mu}$. The first image $I_{\mathcal{C}}$ returns the diffuse colors $\mathcal{C}_R, \mathcal{C}_G, \mathcal{C}_B$ of the scene and the second image $I_{s\alpha\mu}$ encodes the specular component s, the roughness α and the metallicity μ in each of the channels. Examples of $I_{\mathcal{C}}$ and $I_{s\alpha\mu}$ are displayed on the right of Figure 2 of the main document and in Figure 8. Since the CARLA engine does not provide any infrared diffuse color information d, we use the infrared forward model presented by Gruber *et al*. [17] to model it based on the diffuse color image $I_{\mathcal{C}}$ and metallicity μ stored in the $I_{s\alpha\mu}$ image such that

$$d = \mathcal{I}_{\text{NIR}}(I_{\mathcal{C}}) \otimes (1 - \mu), \tag{20}$$

where $\otimes$ denotes a pixelwise product. The $1 - \mu$ factor takes into account that metallic materials with

$\mu = 1$ have no diffuse component ($\rightarrow \ d = 0$). $\mathcal{I}_{\text{NIR}}$ converts the diffuse colors image $I_{\mathcal{C}}$ into a single channel image using

$$\mathcal{I}_{\text{NIR}}(I_{\mathcal{C}}) = (0.299\mathcal{C}_R^{\max} + 0.587\mathcal{C}_G^{\max} + 0.114\mathcal{C}_B^{\max}), \tag{21}$$

with $\mathcal{C}_x^{\max} = \max(\mathcal{C}_x, 1 - \mathcal{C}_x)$ is a pixelwise maximum for each diffuse color channel $x \in \{R, G, B\}$. $\mathcal{I}_{\text{NIR}}(I_{\mathcal{C}})$ is also enhanced with a gamma correction of $\gamma = 0.25$ to darken the image since it is bright by design due to the maximum overlay from $\mathcal{C}_x^{\max}$ [17]. For simplicity, we then encode the s, d and α values into one joint three-channel image $I_{sd\alpha}$.

Correspondences between extracted point clouds and shader images are established through an extrinsic calibration upon camera placement. As such, the values to compute the BRDF can be found by projecting the point cloud into the images and matching LiDAR points with pixel values. The projection procedure is the same to obtain the ambient illumination and is described in Section 2.4.

2.3. Ambient Illumination Model

The wavefront ambient illumination component is modeled as follows

$$a(t) = \Lambda_e(\mathbf{v}) + \int_{\mathbb{S}^+_{-\mathbf{n}}} \Lambda_{\text{ambient}}(\mathbf{l})\text{BRDF}(\mathbf{v}, \mathbf{l}, \mathbf{n})(\mathbf{n} \cdot \mathbf{l})d\mathbf{l}, \tag{22}$$

where t denotes the exposure time of each laser diode. While Eq. (22) is modeled as time-independent in our simulation, in practice, $a(t)$ can be time-dependent if illumination changes at the ns scale. Hence, for clarity, since $a(t)$ is constant in t, we denote by a_i the ambient illumination associated with the point $\mathbf{R}_i$. We note that the ambient illumination for visible light wavelengths is captured by a passive rendered RGB image through the CARLA engine. As such, we adopt the same approach as [60] and approximate Eq. (22) in the NIR band using the red channel of such images, defined as I_{red}, as it is the closest to the NIR band. To capture the RGB image, as for the $I_{sd\alpha}$ images, we set up four cameras around the LiDAR sensor and retrieve the red color channel values for each laser beam through the extrinsic calibration as for the BRDF parameters (s, d, α). The cameras position setup is depicted on the left of Figure 8 and details on projecting point cloud data and retrieving the matching a_i from I_{red} is detailed in Section 2.4.

2.4. Point Cloud Projection

Both the scanned BRDF described in Sec. 2.1 and Sec. 2.2 and the ambient light estimation from fully rendered RGB images described in Sec. 2.3 use images rendered in CARLA for each point cloud frame. Each image is $w = 1280$ pixels wide by $h = 800$ pixels in height and each camera spans a 90° field of view. The cameras are placed such that the whole 360° FOV around the ego vehicle is covered by uniformly distributing them along the azimuth viewing angle as depicted on the left of Fig. 8. All 3D points $\mathbf{R}_i$ from a point cloud $\mathbf{O}$ are projected onto the camera planes using a projection matrix M before rounding down the results to get the corresponding pixel coordinates (u_i, v_i). We note here that we used the native ray casting LiDAR mechanism from the CARLA engine to get both the 3D point coordinates $\mathbf{R}_i$ and the corresponding incident angle to the surface normal θ. This procedure can be formalized by $\mathcal{P}_i(I)$ which extracts information encoded at the aforementioned pixel coordinates on image I, that is

$$\mathcal{P}_i(I) = I[u_i, v_i], \tag{23}$$

where u_i, v_i are the projected pixel coordinates of the point $\mathbf{R}_i$ within image I which are computed using

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \left\lfloor \frac{1}{\tilde{R}_i^z} M \cdot \tilde{\mathbf{R}}_i \right\rfloor, \tag{24}$$

where $\tilde{\mathbf{R}}_i$ is the point $\mathbf{R}_i$ expressed in the camera's frame of reference, $\tilde{R}_i^z$ is the z component of $\tilde{\mathbf{R}}_i$ with $\lfloor x \rfloor$ denoting a componentwise floor operation. The camera projection matrix M is the same for every camera and since we used square camera pixels it is written as

$$M = \begin{bmatrix} \mathscr{F} & 0 & w/2 \\ 0 & \mathscr{F} & h/2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{25}$$

where $\mathscr{F} = w/(2\tan(\text{FOV}))$ is the image plane distance to the camera with FOV denoting the camera field of view. Since the latter is exactly $90°$, then we have $\mathscr{F} = w/2$.

2.5. Multipath Pulse Reflections

LiDAR beams can only be collimated to a limited extent for long ranges. For long distances the beams typically illuminate areas of tens of centimeters. This directly couples object clutter to distance, *e.g.*, a pedestrian limb at long distances might not be discernible from the torso. When analyzing the wavefront, this may lead to increased width of the measured peaks or cause disjoint multiple peaks which can influence the returned point cloud by the LiDAR DSP. In our work, we focus on multiple-echo effects around object discontinuities as they are a typical source of multipath effects in common scenarios like clear weather and in dry environments. To this end, we supersample raycast LiDAR measurements first. We used 128 real LiDAR channels and for each of these channels, we computed 4 extra virtual channels leading to a total of 640 channels in the upsampled point cloud. Similarly, each channel samples the $360°$ horizontal FOV with 875 real points and each of them is accompanied by 4 extra virtual points giving 4375 points per channel in total. Therefore, the maximal number of points in a given upsampled frame is $4375 \times 640 = 2.8$ million points.

The multipath mixing is implemented linearly over all neighboring virtual points and channels into a single waveform. This is described by the $\mathcal{N}(j)$ (resp. $\mathcal{N}(m)$) set of Eq. (8) in the main document as

$$\psi_j^{(m)}(R) = \sum_{i \in \mathcal{N}(j), n \in \mathcal{N}(m)} K_i^{(n)} (H * g^{(n)})_i(R) + a_i, \tag{26}$$

which is the j point itself with its adjacent virtual points (resp. the m channel itself with its adjacent virtual channels). In our simulation, we used $\mathcal{N}(j) = \{j-2, j-1, j, j+1, j+2\}$; similarly for $\mathcal{N}(m)$. In other words, each multi-echo waveform is a linear combination of $|\mathcal{N}(j)| \times |\mathcal{N}(m)| = 25$ single-echo sub-waveforms. In Eq. (26), m and j make leaps of 5 in order not to have beam overlaps, *i.e.*, $j \in \{2, 7, 12, \ldots, 4372\}$ and $m \in \{2, 7, \ldots, 637\}$ with indices starting at 0. Thus, this can be viewed as a strided convolution where each stride step equals the kernel size. We translate unregistered LiDAR points as having an infinite distance which makes their single-echo sub-waveform equal to the ambient illumination component a_i only. For the normalized linear coefficients $K_i^{(n)}$, we define that they describe a Gaussian beam profile where the center of the beam has the highest contribution while the furthest has the least, that is

$$K_i^{(n)} = A \times 2^{-\mathcal{R}(i)^2 - \mathcal{R}(n)^2}, \tag{27}$$

where $\mathcal{R}(z) \equiv z \bmod 5 - 2$ and $A = 1/4.515625$ is the normalization factor such that the sum of $K_i^{(n)}$ over all indices of the same beam is 1.

2.6. Additional Details on Sensing and DSP Model

Environment perception by LiDAR sensors depends on the waveform generation and subsequent peak distance and intensity estimation. Within our simulation pipeline, this translates to point cloud calculations depending not only on the threshold parameter $V^{(m)}$, but on the laser peak power $P_0^{(m)}$ and width $\tau^{(m)}$ as well. Moreover, these parameters are also fed to the DSP in order to reconstruct the distance R, denoted $\Phi^{(R)}(\mathbf{\Theta})$, and scaled target reflectivity ρ/R^2, denoted $\Phi^{(I)}(\mathbf{\Theta})$. Thereby, the designed intensity calibration can have different levels [25]. They range from level zero where uncalibrated intensity values are provided by the sensor to a level three rigorous radiometric calibration providing true scene reflectivies consistent across different sensors. In our work, for a given channel m and point j, we optimize the LiDAR to reconstruct the distance associated with the beam center distance R_j for channel m and the integrated beam intensity

$$I_j^{(m)} = \sum_{i \in \mathcal{N}(j), n \in \mathcal{N}(m)} K_i^{(n)} \frac{C P_0^{(n)} \rho_i}{4 R_i^2}. \tag{28}$$

Assuming a typical rotating scanning setup [20], each channel m is equipped with one dedicated laser diode and detector. Therefore, we assume that the sensing parameters pulse peak power $P_0^{(m)}$ and beam width $\tau^{(m)}$ can be set independently per layer m. For longer distances, especially, an increased power $P_0^{(m)}$ is interesting as it can compensate for power loss due to beam divergence while lower power levels can prevent sensor oversaturation due to strong retroreflectors and short distances. In addition, the Velodyne-style LiDAR sensor [20] is split into two different sub units. This inspired us (see Section 3.5 of the main document) to optimize the sensing parameters for the lower 64- and upper 64-lasers blocks independently. Within each block the parameters $P_0^{(m)}, \tau^{(m)}$ are adjusted via an affine step-function φ of the channel index which can be written as

$$\varphi(m) = \sum_l \omega_l \xi_l(m), \tag{29}$$

where $\omega_l \in \{P_0, \tau\}$ is one of the allowed LiDAR parameter and we sum over all l possible parameters. The value ξ_l determines if a parameter

$$\xi_l(m) = \begin{cases} 1 & \text{if } l \leq s \cdot m + b < l + 1 \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

The thresholds $V^{(m)} \in [0, 2]$ are modeled as tunable continuous parameters which are set to be the same across all channels within a group:

$$V^{(m)} = \begin{cases} V_1 & \text{if } 0 \leq m \leq 63 \\ V_2 & \text{otherwise.} \end{cases} \tag{31}$$

Thus, as seen by the optimizer, we have $\mathbf{\Theta} = \{s_1^{P_0}, s_2^{P_0}, b_1^{P_0}, b_2^{P_0}, s_1^{\tau}, s_2^{\tau}, b_1^{\tau}, b_2^{\tau}, V_1, V_2\}$ where the subscript denotes the group of LiDAR channels. Therefore, the optimizer has a total of 10 knobs to tune in

total. We considered peak power and pulse width as discrete parameters: $P_0^{(m)} \in \{10, 110, 210, \dots 1010\}$ and $\tau^{(m)} \in \{3, 4, \dots, 15\}$ ns. Before feeding the signal $\psi_j^{(m)}$ into the DSP it is discretized into temporal bins $r^{(m)}[k]$ with size $\Delta = 0.2$ ns,

$$r^{(m)}[k] \sim \text{Poisson}\left(\int_{k\Delta}^{(k+1)\Delta} \psi^{(m)}(t)dt\right), \tag{32}$$

where each measured intensity value is Poisson distributed due to the underlying photodiode measuring process [22, 37, 39, 49]. Subsequently, the DSP's $\Phi^{(R,I)}(\mathbf{\Theta})$ task is to identify the dominant scene response and to return the distance R_j and integrated beam intensity $I_j^{(m)}$ from each measured wavefront j and channel m. Firstly, the measurement noise is reduced through a matched filter [54]. The filter uses the same shape and width as the emitted pulse $g^{(m)}$ described by Eq. (4) of the main paper. As the maximal measurement distance is two to three orders of magnitudes higher than the emitted pulse width $\tau^{(m)}c$, the background ambient illumination occupies most of the measurement bins of the waveform. Therefore, the median is a reasonable estimator for the ambient value $a(t)$ which is then subtracted from the captured waveform. The result is then clipped to a lower value of 0. Next, the remaining elevations in the waveform can be seen as returned peaks from the scene. Similar to [29], a rising edge detector with threshold $V^{(m)}$ is used to identify the true scene response, which corresponds to the scene distance R_j of this particular beam m. If multiple peaks are visible we return the distance to the peak with the highest intensity value.

Finally, to extract $I_j^{(m)}$, we divide the estimated peak maximum with the emitted pulse power multiplied by the half pulse width $P_0^{(m)}\tau^{(m)}/2$. Then, both estimated values $\Phi^{(R)}(\mathbf{\Theta})$ and $\Phi^{(I)}(\mathbf{\Theta})$ can be fed into the optimization process to match the ground truth scene's values.

2.7. Model Limitations and Possible Extensions

As mentioned in Section 2 of the main document, a growing body of work uses LiDAR simulation models in order to develop and assess 3D detection pipelines [2, 11, 13, 15, 18, 19, 23, 26, 31, 36, 41, 44, 51, 58, 60]. Our simulation model differs from existing ones in that it includes wavefront sensing and DSP modelling. The closest existing model is the AIODrive [60] dataset as it models beam divergence, SPAD quantization and ambient illumination. Indeed, this work uses similar approaches for the SPAD quantization and ambient illumination (see Equation (32) in Section 2.6 and Equation (22) in Section 2.3). Furthermore, AIODrive's depth convolution approach is similar to our supersampled ray casting approach (see Section 2.5: Equation (26)). However, the only information returned by this dataset related to the PC is the final peak positions and heights whereas the rest of the waveform is not modelled. Additionally, contrary to most simulation engines that model infrared light reflection intensity using a simple Lambertian diffusion model, we instead developed an infrared BRDF based on the CARLA engine's image rendering models (see Section 2.1 and 2.2). Combined with a DSP model, our approach could also be generalized to include more real LiDAR effects into the simulation. This is discussed below.

Motion Artifacts can be modeled for each beam and the accumulated point cloud. Hereby, there is a difference to cameras where the effects are mostly known as motion blur [27]. In cameras a blur kernel mixes neighboring pixels due to long exposure times. Typical mechanical spinning LiDARs (*e.g.*, Velodyne HDL-32E) as used in our simulation pipeline use microsecond exposure times per point for each of their simultaneous scanning lasers. Therefore, the mixing of different beams can be ignored in our simulations. However, for continuous wave LiDARs, this effect is important [62], as those LiDARs

can estimate the relative velocity of objects nearby from measuring directly the doppler shift. This however changes entirely the waveform modeling, as well as the downstream processing.

Rolling Shutter and Motion-Scan Effects occur on system level for the generated point cloud. Typically, LiDARs scan the environment and emit sequentially single laser pulses due to eye safety constraints and limited detector resolution. Since a large body of work [32, 47] related to the removal of those system level scan artifacts from known ego vehicle motion already exists, these temporal sampling artifacts were ignored in our simulation to increase efficiency.

Adverse Weather. Augmentation models have been developed in order to simulate adverse weather effects on real LiDAR point clouds obtained in clear weather, *e.g.*, [18, 19]. One could adapt these methods to modify the raw wavefront generation either at the downsampling step (see Equation (26) in Section 2.6) and/or at the returned echo formulation (Equation (4)-(6) of the main document). Furthermore, as the CARLA engine readily supports wetness texture changes [11], these changes could be directly incorporated through the BRDF (Section 2.1) and the ambient illumination (Section 2.3) equations.

3. Optimization

In this section we review the concept of Pareto optimality and motivate the search for "balanced" MOO solutions. We justify the proposed stable max-rank loss both as a scalarization and as a final selection criterion within the Pareto front, the latter being applicable to any set of hyperparameter vectors for which losses are known and consequently any algorithm. We provide additional details regarding Algorithm 1 from the main document, and we exhaustively list the differences and similarities between the proposed optimization method and related work of Mosleh *et al*. [34] and Robidoux *et al*. [43].

3.1. Pareto Domination and Optimality

Basically, Pareto optimality means "No pain, no gain": A configuration is Pareto-optimal if you cannot make anything better without making something worse.

A configuration is Pareto-dominated if it is possible to improve the value of one loss without making another worse. In the notation of the main document ($\mathcal{L}_l$ stands for the lth loss among the L multiple objectives):

Definition 1 (Pareto Domination) *The hyperparameter vector* $\boldsymbol{\Theta}$ *is* Pareto-dominated *if*

$$\exists \hat{\boldsymbol{\Theta}} : \Big(\exists l \in \{1, \dots, \mathrm{L}\} : \mathcal{L}_l(\hat{\boldsymbol{\Theta}}) < \mathcal{L}_l(\boldsymbol{\Theta})\Big) \quad \text{and} \quad \Big(\forall l \in \{1, \dots, \mathrm{L}\} : \mathcal{L}_l(\hat{\boldsymbol{\Theta}}) \leq \mathcal{L}_l(\boldsymbol{\Theta})\Big). \tag{33}$$

The first clause formalizes "make something better"; the second, "without making anything worse." $\hat{\boldsymbol{\Theta}}$ dominates $\boldsymbol{\Theta}$ if they simultaneously hold.

Pareto domination defines a partial ordering: $\hat{\boldsymbol{\Theta}} < \boldsymbol{\Theta}$ if $\hat{\boldsymbol{\Theta}}$ dominates $\boldsymbol{\Theta}$. The *Pareto front* is the set of minimal elements for this ordering. The elements of the Pareto front, the *Pareto points*, are said to be Pareto-optimal. In other words:

Definition 2 (Pareto Optimality) *The hyperparameter vector* $\boldsymbol{\Theta}$ *is* Pareto-optimal *if*

$$\forall \hat{\boldsymbol{\Theta}} : \Big(\forall l \in \{1, \dots, \mathrm{L}\} : \mathcal{L}_l(\boldsymbol{\Theta}) \leq \mathcal{L}_l(\hat{\boldsymbol{\Theta}})\Big) \quad \text{or} \quad \Big(\exists l \in \{1, \dots, \mathrm{L}\} : \mathcal{L}_l(\boldsymbol{\Theta}) < \mathcal{L}_l(\hat{\boldsymbol{\Theta}})\Big). \tag{34}$$

Loss values are usually only known for a relatively small subset of all valid hyperparameter vectors. Leaving aside loss noisiness issues, we consequently only know for certain when an hyperparameter vector is *not* Pareto-optimal. An hyperparameter vector which is not Pareto-dominated by any of the hyperparameter vectors with known losses may have an unknown dominator. A common abuse of terminology consists of restricting Pareto-optimality to hyperparameters with known losses. The Pareto front is then understood to be the set of hyperparameter vectors *with known losses* which are not dominated by another *with known losses*.

3.2. Balanced MOO Solutions

Because sensing hardware and DSPs are configured with *one* single hyperparameter vector [33,34,36, 40,42,43,53,61], not a set of them, we are not interested in the entire Pareto front. Without some way for users of selecting "the one" from the alternatives, a set of better solutions cannot be the final answer even if each of them is a significant improvement over default or expert-tuned hyperparameters. As seen in Figures 3, 4 and 5, quality solvers generally find many Pareto points (relative to each run's data). Although returning the entire Pareto front allows users to use their own criteria, a sensible default should be provided. Mosleh *et al*. [34] and Robidoux *et al*. [43] use the last Pareto point of an optimization run: In the following, we improve on and generalize this choice. We first identify Pareto points unlikely to be worthy candidates and that consequently should be excluded. Then, we present a selection criterion, applicable to *any* set of hyperparameter vectors with known losses, that naturally excludes them.

There is another, more fundamental reason why the Pareto front is not relevant *in its entirety*. Consider the following setting: An hyperparameter vector $\boldsymbol{\Theta}$ which minimizes one loss—that is, a solution to a single objective optimization (SOO) problem—is automatically in the Pareto front of any MOO problem which includes this loss among its objectives provided there is no hyperparameter vector that ties its minimizing loss value. To see this, note that every $\hat{\boldsymbol{\Theta}} \neq \boldsymbol{\Theta}$ then satisfies the second clause of Eq. (34), while $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$ always satisfies the first. Such single objective minimizers, however, often yield large values for one or more of the *other* loss components. An experiment that illustrates this is discussed in Sec. 1.5. In other words, if losses do not conflict why use multi-objective optimization? For this reason, Pareto-optimal solutions that minimize one loss component are generally not desirable in MOO (*e.g*., detecting one class only, or only optimizing for precision but not recall). Not only does this show that single-loss minimizers are generally not of practical import for genuinely multi-objective problems, this justifies searching for *balanced* MOO solutions, namely solutions for which *all* loss components have "good", or at least as good as possible, values.

We argue here about "good". How do we know which element of the Pareto front is (likely to be) best? We saw that "The Pareto point which minimizes one or other of the losses" is unlikely to be desirable in practice. In the context of scalarizing for optimization with single objective CMA-ES, Mosleh *et al*. [34] proposed the following: A good compromise is one for which every loss value ranks well when compared to the values of the same loss within a relevant population. This choice satisfies the following constraint on the choice of selection criterion, constraint which is automatically satisfied if one uses ranks [21]: Whatever criterion is used should be nondimensional so that scale mismatches between losses not introduce bias. Using this principle within a Chebyshev scalarization [12], that is, a weighted max so that, by design, the scalarization tries to simultaneously keep all ranks low, directly leads to the following generally applicable definition of the *weighted max-rank loss*:

Definition 3 (Weighted Max-Rank Loss) *Let* $\{\boldsymbol{\Theta}^q\}_{q=0}^{\mathrm{Q}}$ *be a collection of hyperparameter vectors, let*

$$\{\boldsymbol{\mathcal{L}}^q = (\mathcal{L}_1(\boldsymbol{\Theta}^q), \ldots, \mathcal{L}_{\mathrm{L}}(\boldsymbol{\Theta}^q))\}_{q=0}^{\mathrm{Q}} \tag{35}$$

be the corresponding values of the loss vectors, let $(\mathrm{w}_1, \ldots, \mathrm{w}_{\mathrm{L}})$ *be a collection of positive weights, and let the rank of* $\boldsymbol{\Theta}^p$ *with respect to the loss component* $\mathcal{L}_l$ *be*

$$\mathcal{R}_l^p = \text{rank of } \mathcal{L}_l(\boldsymbol{\Theta}^p) \text{ within } \{\mathcal{L}_l(\boldsymbol{\Theta}^q)\}_{q=0}^{\mathrm{Q}} \tag{36}$$

with ranks counted from 0 *and loss component value ties resolved by left bisection (per Mosleh* et al.*) or with the proposed stable averaging of the left and right bisection (minus* 1*) ranks, where the average is set to* 0 *whenever the left bisection rank vanishes. Then, the* weighted max-rank loss *of* $\boldsymbol{\Theta}^p$ *with respect to* $\{\boldsymbol{\Theta}^q\}_{q=0}^{\mathrm{Q}}$ *is*

$$\mathcal{M}^p = \max_{l \in \{1,\ldots,\mathrm{L}\}} (\mathrm{w}_l \cdot \mathcal{R}_l^p)\,. \tag{37}$$

Note that if there are no loss value ties, the two max-ranks (left bisection and stable) are equal (subtracting 1 from the right bisection rank, as proposed, so that it starts at 0 when there are no ties even before averaging, is required for this to hold).

The proposed selection criterion is now:

Definition 4 ((Balanced) Pareto Point Selection Criterion) *Choose the Pareto point* $\boldsymbol{\Theta}^p$ *with lowest max-rank loss* $\mathcal{M}^p$ *(computed over the entire run).*

Because any weighted max-rank loss minimizer is either a Pareto point or is dominated by a Pareto point with the same weighted max-rank loss value (this is easy to prove), one loses nothing by restricting the "smallest max-rank loss" search to the Pareto front.

The proposed selection criterion overcomes a significant flaw of "take the last Pareto point of the optimization run": When an optimization method has "restarts" [30], has an exploration pattern which, instead of converging to a single point, tries to refine the boundary of the Pareto front as a whole (maximizing the hypervolume, for example [5]) or that minimizes uncertainty (purposely exploring hyperparameters likely to have poor loss values so as to reduce the likelihood of being trapped in a local minimum), one cannot generally expect the last Pareto point of the run to be the best choice. See Sec. 1.6 for an experiment that manifests this issue and demonstrates the value of the proposed selection criterion.

3.3. Optimization Algorithm

For brevity, the improved seatbelting of the proposed method is omitted from the version of Algorithm 1 presented in the main document.

Algorithm 1 follows the standard structure of a $(\mu/\mu_{\mathrm{W}}, \lambda)$-CMA-ES (Covariance Matrix Adaptation-Evolution Strategy) method [21] without restarts [30] as implemented, for example, in the DEAP library [14], with the major difference that losses are evaluated for the centroid of every generation (Line 20 of Algorithm 1) instead of for random samples only (Lines 22 and 25). This is why $4\mathrm{P}+1$ loss evaluations are performed (Lines 20–26), P being the number of hyperparameters being optimized (the dimension of $\boldsymbol{\Theta}$), even though the standard CMA-ES update (Line 32, see [21] and for example the DEAP implementation) only uses $4\mathrm{P}$ children. The extra child—namely $\boldsymbol{\Theta}^{0,n}$, the weighted centroid

Algorithm 1 Lidar Hyperparameter Optimization (Unabridged).

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 [43]
    $\mathbf{\Theta}^{p,n} \leftarrow \mathbf{\Theta}^{p,n}$ 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" [34] centroid weights with $\lambda = 4\mathrm{P}$, $\mu = 3\mathrm{P}$
  if n is odd **then**
    Use "stable" [43] centroid weights with $\lambda = \mu = 4\mathrm{P}$
  end if
  Standard CMA-ES update [21] 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 $\min\limits_{p\in\{0,\ldots,4\mathrm{P}\}} \mathcal{M}^{p,n,n} < \min\limits_{q\in\{0,\ldots,4\mathrm{P}\},m\in\{1,\ldots,n\}} \mathcal{M}^{q,m,n}$ **then**
    $\mathbf{\Theta}_{\text{center}} \leftarrow$ minimizer closest to centroid of minimizers (distance ties resolved arbitrarily)
  end if
end for
return $\mathbf{\Theta}^{p,n}$ in the (guaranteed nonempty) intersection of the Pareto front and the set of minimizers of $\mathcal{M}^{q,m,N}$, with ties resolved by choosing the one closest to their centroid and remaining ties resolved by maximizing n, then p

normally only used as center of the next generated anisotropic Gaussian cloud—is used to provide the greedy CMA-ES branch with one extra, better, candidate, thus improving transients and stability. This proposed greedy variant of CMA-ES works as follows: If any hyperparameter vector of the generation, $\boldsymbol{\Theta}^{0,n}$ included, is a new "best so far," that is, is a strict minimizer of the scalarized loss (Line 34), it replaces the weighted centroid (Line 33) as center of the next generated Gaussian cloud (Line 35).

Such jumps of the random Gaussian cloud center could corrupt the standard CMA-ES path statistics used to evolve the matrix $\mathbf{C}$ (the CMA-ES "directional" covariance matrix factor) and σ (the "scale" factor of the square root of the covariance matrix) because they assume random Gaussian clouds [21] and including the cloud's center would break randomness. Mosleh *et al*. ignore this issue. Robidoux *et al*. prevent spurious jumps in the statistics by resetting path variables to 0 whenever a greedy jump occurs. This, unfortunately, renders the path machinery impotent when many "best so far" are found in quick succession, as sometime happens late in the run when performing MOO with a large number of objectives and hyperparameters. Instead, the proposed method updates the path accumulators *first* (Line 32), *then* moves the Gaussian cloud center to the greedy location (Line 33). Instead of literal path accumulators, the proposed method consequently uses $\mathbf{p}$ and $\mathbf{c}$ to cumulate statistics about promising single steps which are not necessarily taken because possibly overridden by a jump.

Another novel aspect of the proposed method is that, even though generation size is fixed, two different sets of centroid weights are used. Both sets are, in CMA-ES parlance, *active* [1, 21]. As seen in Lines 28–31 of Algorithm 1, the very first generation uses the boundary-stable weights of Robidoux *et al*. without discard, meaning that μ, the number of individuals within the generation with vanishing weights before redistribution to account for ties, is equal to $\lambda = 4\mathrm{P}$. The boundary-stable weights of Robidoux *et al*. alternate with the eager weights of Mosleh *et al*., which use $\mu = 3\mathrm{P}$ and $\lambda = 4\mathrm{P}$. As Nishida *et al*. [35] point out, larger generations are empirically known to be preferable with rough or noisy losses. The reason for using $\mu = \lambda$, that is, no discard, with the boundary-stabilizing weights instead of the alternatives documented in [43] is that, with active CMA-ES methods—methods with centroid weights, some of them negative, monotone as a function of the rank, that discard, as is standard, the worst individuals of the generation—the solver is likely to explore in the wrong direction near generic local minima because the worst directions implicitly gets vanishing centroid weights while less "bad" directions get strictly negative ones. This choice makes it less likely that the boundary-stable generation get lost near minima. In contrast, discarding exactly one quarter of each generation allows Mosleh *et al*. to exploit the symmetry between the second and third quartiles of Gaussian distributions to obtain a robust gradient approximation. Alternating the two types of active weights perform the best in all of our experiments.

For reference, the "eager" weights of Mosleh *et al*. are defined as follows when $\lambda = 4\mathrm{P}$ and $\mu = 3\lambda/4$, as is the case here. With the non-normalized weights

$$\hat{\mathrm{w}}_l = \begin{cases} 2\mathrm{P} - \frac{1}{2} - l & \text{if } l \in \{0, 1, \ldots, 3\mathrm{P} - 1\}\,, \text{ and} \\ 0 & \text{if } l \in \{3\mathrm{P}, \ldots, 4\mathrm{P} - 1\}\,, \end{cases} \tag{38}$$

the eager weights, assuming that there are no ties within the generation (otherwise weights are averaged between ties) are

$$\mathrm{w}_l = \hat{\mathrm{w}}_l / \sum_{k=0}^{3\mathrm{P}-1} \hat{\mathrm{w}}_k \tag{39}$$

so that they sum to 1. The "boundary stable" weights of Robidoux *et al*. are defined similarly: With the non-normalized weights

$$\breve{\mathrm{w}}_l = 1 - \sqrt{2}\frac{l}{4\mathrm{P}-1} \text{ for } l \in \{0, 1, \ldots, 4\mathrm{P}-1\}, \tag{40}$$

the boundary-stable weights, assuming that there are no ties within the generation, are

$$\mathrm{w}_l = \breve{\mathrm{w}}_l / \sum_{k=0}^{4\mathrm{P}-1} \breve{\mathrm{w}}_k. \tag{41}$$

One major improvement over standard CMA-ES which the proposed method adapted from Mosleh *et al*. and Robidoux *et al*. is the use of a tracking Monte Carlo simulation which better accounts for deviations from randomness than χ_n statistics given both hyperparameter value quantization and the impact of boundary conditions.

We now discuss the particulars of the seatbelting of the proposed method (Lines 4–19 and 32). Note that σ scales like the square root of the covariance matrix $\mathbf{C}$. We use $1/3$ as an appropriate upper bound for σ (Lines 6 and 9 of Algorithm 1) because, since we optimize in the unit hypercube $[0,1]^{\mathrm{P}}$, this implies that at most about $(100\% - 99.73\%) \div 2 = 0.135\%$ of the Gaussian cloud $\boldsymbol{\Theta}$s are reflected more than once by the mirroring boundary conditions when $\mathbf{C}$ is close to the identity matrix.

To prevent quantized hyperparameter values from getting stuck in the basin of attraction of an integer, we add Gaussian noise that scales like the distance between the images of integers within the unit interval (Line 23). In addition, ε, the minimum allowed σ and eigenvalue of $\sigma\mathbf{C}^{\frac{1}{2}}$, is kept fairly large $(4/255)$ to tilt the convergence vs. exploration balance slightly toward exploration.

CMA-ES directional search directions are scaled by $\sigma\mathbf{C}^{\frac{1}{2}}$. Because the diameter of the unit hypercube $[0,1]^{\mathrm{P}}$ is $\sqrt{\mathrm{P}}$ and consequently can attain large values (this is the so-called "Curse of Dimensionality"), a somewhat large upper bound on $\sigma\mathbf{C}^{\frac{1}{2}}$'s eigenvalues, namely $\Lambda = \sqrt{\mathrm{P}}/3$, is used so as to allow the search to reach the corners of the search space within a reasonable number of iterations (Lines 1 and 17–19).

We also propose ensuring that $\mathbf{C}^{\frac{1}{2}}$ has at least one eigenvalue no smaller than 1 as well as one no larger than 1 so that the above bounds make sense. If this does not hold, we simultaneously rescale σ and $\mathbf{C}^{\frac{1}{2}}$ to make it true while keeping their product unchanged (Lines 5–7 and 14–16). Combined seatbelting of σ and $\mathbf{C}^{\frac{1}{2}}$ through their product is a novel feature of the proposed method.

Whenever an eigenvalue of $\mathbf{C}$ is too small (because of the active weights, eigenvalues can be negative) or too large, the proposed method replaces $\mathbf{C}$ by its square root before clamping the eigenvalues, the square root being a somewhat arbitrary choice which does the job of pulling $\mathbf{C}$ toward the identity while retaining more information about preferred search directions than hard clamping. Existing CMA-ES methods just clamp. Finally, the $4/3$ multiplier appears in Lines 9 and 32 owing to the fact that with eager weights $\mu = 3\lambda/4$.

3.4. Joint Optimization of the LiDAR and the Object Detection CNN

Robidoux *et al*. [43] jointly optimized a hardware HDR (High Dynamic Range) camera sensor, an ISP operating in HDR mode, and a CNN for object detection and classification by block coordinate descent. A similar approach could be used in the present context. This is a future research direction.

References

[1] Dirk V. Arnold and Nikolaus Hansen. Active covariance matrix adaptation for the (1+1)-CMA-ES. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 385–392, 2010. 31

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences, Aug. 2019. arXiv:1904.01416 [cs]. 26

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. 5, 6, 7

[4] James Bergstra, Daniel Yamins, and David Cox. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *Proceedings of the 30th International Conference on Machine Learning*, pages 115–123. PMLR, Feb. 2013. ISSN: 1938-7228. 5, 6, 7

[5] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, Sept. 2007. 8, 9, 10, 11, 13, 29

[6] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, Oct. 2016. Conference Name: IEEE Transactions on Evolutionary Computation. 8, 9, 10, 11, 12

[7] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM SIGGRAPH Computer Graphics*, 15(3):307–316, Aug. 1981. 20

[8] Kalyanmoy Deb. Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. In Lihui Wang, Amos H. C. Ng, and Kalyanmoy Deb, editors, *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pages 3–34. Springer, London, 2011. 11

[9] K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms | Proceedings of the 8th annual conference on Genetic and evolutionary computation. 8, 9, 10, 11, 14

[10] Kalyanmoy Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 635–642, New York, NY, USA, July 2006. Association for Computing Machinery. 8, 9, 10, 11, 14

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 13–15 Nov 2017. 21, 22, 26, 27

[12] Michael T. M. Emmerich and André H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, Sept. 2018. 5, 28

[13] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented LiDAR Simulator for Autonomous Driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, Apr. 2020. arXiv:1811.07112 [cs]. 26

[14] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012. 29

[15] Christopher Goodin, Daniel Carruth, Matthew Doude, and Christopher Hudson. Predicting the influence of rain on LiDAR in ADAS. *Electronics*, 8, 2019. 26

[16] Tobias Gruber, Mario Bijelic, Felix Heide, Werner Ritter, and Klaus Dietmayer. Pixel-accurate depth evaluation in realistic driving scenarios. In *2019 International Conference on 3D Vision (3DV)*, pages 95–105. IEEE, 2019. 16

[17] Tobias Gruber, Frank Julca-Aguilar, Mario Bijelic, and Felix Heide. Gated2depth: Real-time dense lidar from gated images. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 22, 23

[18] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. LiDAR Snowfall Simulation for Robust 3D Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 26, 27

[19] Martin Hahner, Christos Sakaridis, Dengxin Dai, and Luc 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. 26, 27

[20] Ryan Halterman and Michael Bruch. Velodyne HDL-64E lidar for unmanned surface vehicle obstacle detection. In *Unmanned Systems Technology XII*, volume 7692, pages 123–130. SPIE, May 2010. 25

[21] Nikolaus Hansen. The CMA evolution strategy: A tutorial, 2016. 5, 28, 29, 30, 31

[22] Felix Heide, Steven Diamond, David B Lindell, and Gordon Wetzstein. Sub-picosecond photon-efficient 3d imaging using single-photon sensors. *Scientific reports*, 8(1):1–8, 2018. 26

[23] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception, May 2019. arXiv:1905.00160 [cs]. 26

[24] James T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, Aug. 1986. Association for Computing Machinery. 19

[25] Alireza Kashani, Michael Olsen, Christopher Parrish, and Nicholas Wilson. A review of LiDAR radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11), 2015. 25

[26] Velat Kilic, Deepti Hegde, Vishwanath Sindagi, A. Brinton Cooper, Mark Foster, and Vishal Patel. LiDAR light scattering augmentation (LISA): physics-based simulation of adverse weather conditions for 3D object detection. *arXiv preprint 2107.07004*, 2021. 26

[27] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018. 26

[28] Ke Li, Renzhi Chen, Guangtao Fu, and Xin Yao. Two-Archive Evolutionary Algorithm for Constrained Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315, Apr. 2019. Conference Name: IEEE Transactions on Evolutionary Computation. 8, 9, 10, 11, 13

[29] Xiaolu Li, Bingwei Yang, Xinhao Xie, Duan Li, and Lijun Xu. Influence of waveform characteristics on lidar ranging accuracy and precision. *Sensors*, 18(4), 2018. 26

[30] Ilya Loshchilov. CMA-ES with restarts for solving CEC 2013 benchmark problems. In *2013 IEEE Congress on Evolutionary Computation*, pages 369–376, 2013. 29

[31] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11164–11173, 2020. 26

[32] Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur, and Xavier Savatier. LiDAR point clouds correction acquired from a moving car based on CAN-bus data, June 2017. arXiv:1706.05886 [cs]. 27

[33] Anish Mittal, Anush K. Moorthy, and Alan C. Bovik. Automatic parameter prediction for image denoising algorithms using perceptual quality features. In *Human Vision and Electronic Imaging XVII*, volume 8291, pages 110–116. SPIE, Feb. 2012. 28

[34] Ali Mosleh, Avinash Sharma, Emmanuel Onzon, Fahim Mannan, Nicolas Robidoux, and Felix 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)*, pages 7526–7535, Seattle, WA, USA, June 2020. IEEE. 5, 6, 7, 8, 9, 10, 11, 13, 27, 28, 30

[35] Kouhei Nishida and Youhei Akimoto. Population Size Adaptation for the CMA-ES Based on the Estimation Accuracy of the Natural Gradient. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 237–244, New York, NY, USA, July 2016. Association for Computing Machinery. 31

[36] Jun Nishimura, Timo Gerasimow, Rao Sushma, Aleksandar Sutic, Chyuan-Tyng Wu, and Gilad Michael. Automatic isp image quality tuning using nonlinear optimization. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2471–2475, 2018. 26, 28

[37] Matthew O'Toole, Felix Heide, David B. Lindell, Kai Zang, Steven Diamond, and Gordon Wetzstein. Reconstructing Transient Images from Single-Photon Sensors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2289–2297, Honolulu, HI, July 2017. IEEE. 26

[38] Annibale Panichella. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 595–603, New York, NY, USA, July 2019. Association for Computing Machinery. 8, 9, 10, 11, 12

[39] Adithya K. Pediredla, Aswin C. Sankaranarayanan, Mauro Buttafava, Alberto Tosi, and Ashok Veeraraghavan. Signal Processing Based Pile-up Compensation for Gated Single-Photon Avalanche Diodes, June 2018. arXiv:1806.07437 [physics]. 26

[40] Luke Pfister and Yoram Bresler. Learning Filter Bank Sparsifying Transforms. *IEEE Transactions on Signal Processing*, 67(2):504–519, Jan. 2019. Conference Name: IEEE Transactions on Signal Processing. 28

[41] Francesco Pittaluga, Zaid Tasneem, Justin Folden, Brevin Tilmon, Ayan Chakrabarti, and Sanjeev J. Koppal. Towards a MEMS-based Adaptive LIDAR, Oct. 2020. arXiv:2003.09545 [cs, eess]. 26

[42] Geoffrey Portelli and Denis Pallez. Image signal processor parameter tuning with surrogate-assisted particle swarm optimization. In Lhassane Idoumghar, Pierrick Legrand, Arnaud Liefooghe, Evelyne Lutton, Nicolas Monmarché, and Marc Schoenauer, editors, *Artificial Evolution - 14th International Conference, Évolution Artificielle, EA 2019, Mulhouse, France, October 29-30, 2019, Revised Selected Papers*, volume 12052 of *Lecture Notes in Computer Science*, pages 28–41. Springer, 2019. 28

[43] Nicolas Robidoux, Luis E. Garcia Capel, Dong-eun Seo, Avinash Sharma, Federico Ariza, and Felix Heide. End-to-End High Dynamic Range Camera Pipeline Optimization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6297–6307, 2021. 7, 27, 28, 30, 31, 32

[44] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio 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)*, pages 3234–3243, June 2016. ISSN: 1063-6919. 26

[45] Christophe Schlick. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, 13(3):233–246, 1994. 21

[46] Haitham Seada and Kalyanmoy Deb. A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives. *IEEE Transactions on Evolutionary Computation*, 20(3):358–369, June 2016. Conference Name: IEEE Transactions on Evolutionary Computation. 8, 9, 10, 11, 14

[47] Mao Shan, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Probabilistic egocentric motion correction of lidar point cloud and projection to camera images for moving platforms. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020. 27

[48] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 16

[49] Dongeek Shin. *Computational imaging with small numbers of photons*. Thesis, Massachusetts Institute of Technology, 2016. Accepted: 2016-07-18T20:05:44Z. 26

[50] B. Smith. Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation*, 15(5):668–671, Sept. 1967. Conference Name: IEEE Transactions on Antennas and Propagation. 21

[51] Tao Sun, Mattia Segu, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari, and Fisher 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)*, pages 21371–21382, June 2022. 26

[52] T. S. Trowbridge and K. P. Reitz. Average irregularity representation of a rough surface for ray reflection. *JOSA*, 65(5):531–536, May 1975. Publisher: Optica Publishing Group. 21

[53] Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl ST. Arnaud, Derek Nowrouzezahrai, Jean-François Lalonde, and Felix Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Transactions on Graphics*, 38(4):1–14, Aug. 2019. 28

[54] G. Turin. An introduction to matched filters. *IRE Transactions on Information Theory*, 6(3):311–329, 1960. 26

[55] Peter O'Connor (https://github.com/petered). Artemis. https://github.com/QUVA-Lab/artemis/tree/peter, 2018. 9

[56] Lucien S. (https://stackoverflow.com/users/1208142/lucien-s). Fast calculation of pareto front in python. URL: https://stackoverflow.com/q/32791911/6362595 (version: 2023-03-24). 9

[57] Yash Vesikar, Kalyanmoy Deb, and Julian Blank. Reference Point Based NSGA-III for Preferred Solutions. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1587–1594, Nov. 2018. 8, 9, 10, 11, 14

[58] Niclas Vödisch, Ozan Unal, Ke Li, Luc Van Gool, and Dengxin Dai. End-to-End Optimization of LiDAR Beam Configuration for 3D Object Detection and Localization. *IEEE Robotics and Automation Letters*, 7(2):2242–2249, Apr. 2022. Conference Name: IEEE Robotics and Automation Letters. 26

[59] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 195–206, 2007. 21

[60] Xinshuo Weng, Yunze Man, Jinhyung Park, Ye Yuan, Dazhi Cheng, Matthew O'Toole, and Kris Kitani. All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds. *arXiv*, 2021. 23, 26

[61] Chyuan-Tyng Wu, Leo F. Isikdogan, Sushma Rao, Bhavin Nayak, Timo Gerasimow, Aleksandar Sutic, Liron Ain-kedem, and Gilad Michael. VisionISP: Repurposing the image signal processor for computer vision applications. In *IEEE International Conference on Image Processing (ICIP)*, pages 4624–4628, 2019. 28

[62] Ji Zhang and Sanjiv Singh. LOAM : LiDAR odometry and mapping in real-time. *Robotics: Science and Systems Conference (RSS)*, 2014. 26