Hardware-in-the-loop End-to-end Optimization of Camera Image Processing Pipelines (Supplemental Material)

Ali Mosleh[1] Avinash Sharma[1] Emmanuel Onzon[1]
Fahim Mannan[1] Nicolas Robidoux[1] Felix Heide[1,2]

[1]Algolux [2]Princeton University

1. Details on ISPs used for Assessment

1.1. ARM Mali-C71 ISP

The ARM Mali-C71 ISP [2] is a state-of-the-art hardware ISP for real-time applications in consumer devices, robotics, and automotive imaging. Its processing stages include *demosaicking*, *white balancing*, *denoising*, *edge enhancement*, *gamma and tone control* [2]. The hyperparameters of this ISP span a 31-dimensional space that we optimize for specific downstream tasks, for example, to process RAW pixel data so it can be displayed, or used for further processing by computer vision algorithms. Table 1 lists the ARM Mali-C71 ISP hyperparameters modified in our experiments. Three of the thirty-one parameters are associated with white balance. These were kept fixed during our object detection and perceptual image quality optimization experiments (Sec. 6.1 and 6.3 of the main document). A preliminary color calibration was performed before the experiments and the white balance gains were kept constant thereafter. Therefore, optimization experiments only involved the remaining 28 hyperparameters.

Table 1: ARM Mali-C71 ISP hyperparameters and their operational ranges. Hyperparameters are discrete; they are relaxed to continuous values in the optimization process.

Denoising

Parameter	Operational Range
thresh 1h	$\{0, \ldots, 255\}$
strength 1	$\{0, \ldots, 255\}$
thresh 4th	$\{0, \ldots, 255\}$
thresh long	$\{0, \ldots, 255\}$

White Balance

Parameter	Operational Range
gain 00	$\{0, \ldots, 4095\}$
gain 01	$\{0, \ldots, 4095\}$
gain 11	$\{0, \ldots, 4095\}$

Color Space Conversion

Parameter	Operational Range
coef a 11	$\{0, \ldots, 5880\}$
coef a 12	$\{0, \ldots, 5880\}$
coef a 21	$\{0, \ldots, 5880\}$
coef a 22	$\{0, \ldots, 5880\}$
coef a 31	$\{0, \ldots, 5880\}$
coef a 32	$\{0, \ldots, 5880\}$

Color & Tone Correction

Parameter	Operational Range
lut knee	$\{0, \ldots, 255\}$
lut power	$\{0, \ldots, 255\}$
lut shadow	$\{0, \ldots, 255\}$

Demosaicking

Parameter	Operational Range
vh slope	$\{0, \ldots, 255\}$
vh thresh	$\{0, \ldots, 4095\}$
va slope	$\{0, \ldots, 255\}$
va thresh	$\{0, \ldots, 4095\}$
aa slope	$\{0, \ldots, 255\}$
aa thresh	$\{0, \ldots, 4095\}$
uu slope	$\{0, \ldots, 255\}$
uu thresh	$\{0, \ldots, 4095\}$
sharp alt ld	$\{0, \ldots, 255\}$
sharp alt ldu	$\{0, \ldots, 255\}$
sharp alt lu	$\{0, \ldots, 255\}$
fc alias slope	$\{0, \ldots, 255\}$
fc alias thresh	$\{0, \ldots, 255\}$
fc slope	$\{0, \ldots, 255\}$
np offset	$\{0, \ldots, 255\}$

1.2. OnSemi AP0202AT ISP

The AP0202AT ISP is a state-of-the-art automotive ISP targeting high dynamic range (HDR) sensors. Its processing stages include *defect pixel correction*, *demosaicking*, *denoising*, *black level adjustment*, *tone mapping*, *color correction* and *aperture correction* [1]. Table 2 lists the fourteen OnSemi AP0202AT ISP hyperparameters modified by our experiments. Although this ISP is vendor-optimized to provide "full auto-functions support" and thus its hyperparameter space is considerably smaller than most ISPs, further optimization for the target end-use task is still required, as shown in Sec. 6.3 of the main document.

Table 2: OnSemi AP0202AT ISP hyperparameters and their operational ranges. Hyperparameters are discrete; they are relaxed to continuous values in the optimization process.

Aperture Correction

Parameter	Operational Range
ap gain bright	$\{0, \ldots, 255\}$
ap thresh high	$\{0, \ldots, 255\}$
ap gain dark	$\{0, \ldots, 255\}$
ap thresh dark	$\{0, \ldots, 255\}$

Gamma Correction

Parameter	Operational Range
contrast gradient bright	$\{0, \ldots, 255\}$
contrast intercept point bright	$\{0, \ldots, 255\}$
contrast gradient dark	$\{0, \ldots, 255\}$
contrast intercept point dark	$\{0, \ldots, 255\}$

Demosaicking

Parameter	Operational Range
demosaic low	$\{0, \ldots, 255\}$
demosaic high	$\{0, \ldots, 255\}$

Denoising (Adaptive Color Difference)

Parameter	Operational Range
adacd gr weights strength low	$\{0, \ldots, 16\}$
adacd gr weights strength high	$\{0, \ldots, 16\}$

1.3. Synthetic ISP

Given that a large number of images are required to evaluate computer vision performance, and following [6, 7, 9, 24], we developed a synthetic software ISP in order to speed up prototyping and evaluation. This synthetic ISP consists of a sequence of processing blocks that perform *white balancing*, *demosaicking*, *denoising*, *sharpening*, *tone and color correction*, and *compression*[1].

The synthetic ISP is used in the experiments of Sec. 6.1, 6.2 and 6.4 of the main document. The processing blocks optimized in these experiments are the denoising, sharpening, and tone and color correction blocks. A summary of their operation, with a description of the hyperparameters optimized by the proposed method, follows:

Denoiser The denoiser block includes 3 different methods;

- *Luminance NLM Denoiser*: A non-local means (NLM) filtering algorithm is applied to the luminance component of the image with patch size 'nlm templateWindowSize', patch search distance 'nlm searchWindowSize' in pixels, and filter strength 'nlm h' [8].
- *Trichromatic Gaussian Denoiser*: A Gaussian filter G_σ is convolved ($*$) with each of three color channels to suppress high-frequency noise in the input image I, *i.e.*, $I * G_\sigma$. The standard deviation σ of the Gaussian filter can be varied with the hyperparameter 'gaussian sigma'.
- *Chroma Denoiser*: "Blotchy" color noise artifacts that appear in the image are treated as low-frequency chrominance features. They are removed with an approximation of Laplacian-of-Gaussian filtering [13]. Specifically, let G_{σ_1} and G_{σ_2} ($\sigma_2 > \sigma_1$) denote two different Gaussian kernels with standard-deviations 'chroma sigma1' and 'chroma sigma2', and let k denote a scalar 'chroma gain'. The chroma denoiser is applied to the chrominance I of the image giving $I - k\,|I * G_{\sigma_1} - I * G_{\sigma_2}|$.

Sharpening Unsharp masking [13] is applied on the luminance component of the image to enhance edges. It replaces the luminance I by $I + k\,(I - I * G_\sigma)$, where G_σ denotes a Gaussian kernel with standard-deviation σ 'unsharp sigma', and the gain control parameter 'unsharp strength' is denoted by k.

[1] ISP and RAW data provided by the authors upon request.

Tone and Color Correction The tone and color correction block has two different blocks:

- *Contrast Stretching*: A sigmoid function [11] is applied to shift the characteristic curve along the horizontal direction with the cutoff parameter 'contrast knee' and the gain parameter 'contrast gain' which serves as the constant multiplier in the exponential in the sigmoid function. Let g and c denote 'contrast gain' and 'contrast knee', respectively. Contrast stretching is applied on the input image pixel x as $1/\left(1+\exp\left(g(c-x)\right)\right)$.
- *Tone Mapping*: This tone mapper maps each color channel, normalized to the interval $\mathbb{R}_{[0,1]}$, independently as follows. It transforms $x \in \mathbb{R}_{[0,1]}$, the value of the color channel before being modified by the tone mapper, into

$$x^{\frac{1}{\gamma}} \quad \text{where} \quad \frac{1}{\gamma} = \frac{1}{\gamma_1} \cdot \frac{1-(1-\gamma_2)\, x^{\frac{1}{\gamma_1}}}{1-(1-\gamma_2)\, k^{\frac{1}{\gamma_1}}}.$$

  $\gamma_1 \geq 1$ is a gamma correction that mostly affects highlights, $\gamma_2 \leq 1$ is a gamma correction that mostly affects shadows, and $0 < k < 1$ is a knee, specifically, k is the fixed point of the transformation when $\gamma_1 = 1$. (The actual fixed point is left of the value of k then $\gamma_1 > 1$.) The corresponding synthetic ISP hyperparameters are called 'gamma gamma1', 'gamma gamma2' and 'gamma knee'. Their useful ranges are remapped to 16-bit integers.

Table 3 lists the synthetic ISP hyperparameters optimized in our experiments. Note that, like the hyperparameters of the hardware ISPs which the synthetic ISP crudely models, the hyperparameters of the synthetic ISP only take discrete values. As explained in Sec. 3 of the main document and Sec. 2.1 below, these discrete values are relaxed to continuous real numbers and then normalized to the interval $\mathbb{R}_{[0,1]}$ within the optimizer.

Table 3: Hyperparameters of the synthetic ISP and their operational ranges. Like hardware ISPs, the synthetic ISP takes discrete hyperparameters by design, but we map them to continuous space in the optimization process.

Denoising

Parameter	Operational Range
nlm templateWindowSize	$\{0,\ldots,13\}$
nlm searchWindowSize	$\{0,\ldots,31\}$
nlm h	$\{0,\ldots,14\}$
gaussian sigma	$\{0,\ldots,4095\}$
chroma sigma1	$\{0,\ldots,4095\}$
chroma sigma2	$\{0,\ldots,4095\}$
chroma gain	$\{0,\ldots,4095\}$

Tone and Color Correction

Parameter	Operational Range
contrast knee	$\{0,\ldots,4095\}$
contrast gain	$\{0,\ldots,255\}$
gamma knee	$\{0,\ldots,65535\}$
gamma gamma1	$\{0,\ldots,65535\}$
gamma gamma2	$\{0,\ldots,65535\}$

Sharpening

Parameter	Operational Range
unsharp sigma	$\{0,\ldots,4095\}$
unsharp strength	$\{0,\ldots,255\}$

2. Detailed Description of the Proposed Solver

The proposed solver has two main stages: Search Space Reduction, an exploration/coarse convergence stage, and CMA-ES (Covariance Matrix Adaptation-Evolution Strategy [17, 18, 15, 4, 16]), a refinement/fine convergence stage. Stages can be repeated, together or individually. Both are stochastic. As is common with stochastic and evolutionary search methods, several algorithmic details are motivated by heuristics.

Many times in the following discussion, a "best so far" hyperparameter setting is mentioned. Generally, the *best so far* is the best known hyperparameter setting for the image capture conditions and downstream task under consideration. At the start of optimization with the proposed method to produce the results presented in this paper, the initial search point and best so far was invariably taken to be the ISP vendor default. Later in the optimization process, the best so far is inherited from previous stages. Specifically, the best so far hyperparameter setting is selected based on its max-rank loss, a novel loss construction for Multi-Objective Optimization (MOO) which we now motivate and explain in detail.

2.1. Continuous Relaxation of Native Hyperparameter Values

As mentioned in the main document, we relax native, integer, ISP hyperparameter values to continuous values in $\mathbb{R}_{[0,1]}$. When we optimize a native ISP hyperparameter θ_p over the range of discrete integer values

$$\{\text{lowest value}, \text{lowest value}+1, \ldots, \text{lowest value}+\text{B}-1\},$$

the corresponding relaxed hyperparameter in the unit interval is

$$\Theta_p = \frac{\theta_p - \text{lowest value}}{\text{B}-1}.$$

This affine transformation maps the lowest native ISP hyperparameter value under consideration to 0, and the highest to 1.

2.2. Max-rank Loss Scalarization

Scalarization is the process of converting a multi-objective loss vector into a single, scalar, loss. Scalarization thus allows single objective optimizers to tackle MOO problems. The most common scalarization is weighted convex combination, in which the scalar loss is a weighted sum, with positive weights, of the vector loss components. The proposed hardware-in-the-loop optimization method uses a different, novel, scalarization: the *(dynamically) weighted max-rank loss*.

2.2.1 Motivation

Recall that a *trial* is an hyperparameter setting for which the loss is computed and thus known. Also recall that when performing MOO, the loss has several components, each derived from one or more task-specific evaluation metric. In principle, each of the loss components is to be made as low as possible; only in principle however, for the following reason. Loss components are often in opposition. For one thing, if all losses were in agreement, we could optimize using only one. Using more than one evaluation metric indicates that multiple ways of ranking results are being considered. In fact, losses that address different, possibly conflicting, aspects of performance are often used. For example, a loss component that only sees noise is likely to be inversely correlated with one that only sees texture. Generally, when optimizing for perceptual quality, we aim for output images which have both low noise *and* a lot of texture: No noise and no texture, or lots of texture and lots of noise, are considered unacceptable. Precision and recall are another example: perfect precision but very poor recall, or vice versa, is generally considered unacceptable.

In summary: When performing MOO, we often desire "balanced" solutions; Getting good values for some but not all evaluation metrics is generally not enough.

Thus, when performing MOO, we aim to find a Pareto point which is a good compromise between conflicting objectives. The dynamically weighted max-rank loss scalarization was designed to guide a single objective optimizer toward such a solution. Leaving aside the dynamic aspect of the weighting, the weighted max-rank loss is the weighted Chebyshev Scalarization [12] computed after converting loss components to ranks. Because it involves ranks and, consequently, other trials, the max-rank loss is an optimization device: For a given hyperparameter setting, the max-rank loss scalarization is only defined with respect to a population: the trials that enter in the ranking. Thus, unlike typical loss functions (PSNR, SSIM, etc.), the max-rank loss is not an absolute quantity: the value of the max-rank loss is *relative* to the loss values of a group of trials. Accordingly, let us call a group of trials with respect to which ranks are computed a *reference population*.

Being rank-based, both the proposed Search Space Reduction and CMA-ES are invariant with respect to strictly monotone transformations of loss function values. This is a well-known property of CMA-ES. For example, CMA-ES optimization with respect to RMSE is the same as CMA-ES optimization with respect to PSNR. When performing single objective optimization with such methods, there is in principle no difference between optimizing directly with respect to the loss ("as is") and optimizing with respect to weighted max-rank loss, because they order trials exactly the same way. (Weighting does not impact single objective optimization provided the weight is fixed within the ranking, which it is, and provided the weight is never 0, which is trivially enforced.) Consequently, switching to max-rank loss only impacts single objective optimization when the optimizer is *not* invariant with respect to monotone loss value transformations (most gradient-based methods are not). Switching to max-rank loss however almost invariably impacts multi-objective optimization.

Performing MOO using max-rank scalarization instead of, *e.g.*, the commonly used convex combination scalarization, has several advantages. The first is that it provides a natural normalization of the different losses, that is, it removes the need for the loss components to be rescaled so that they have commensurate responses. Without conversion to ranks, the needed rescaling generally depends on capture conditions. In addition, such rescalings are often folded into the loss weights themselves. The weights then serve double duty: communicating the desired relative importance of loss components to the solver; and rescaling the loss components so that they are commensurate. This obfuscates their impact. Consequently, the rescaling, which not only needs to make ranges of values commensurate but also their variations (their "gradients"), often needs to be changed when capture conditions and other context specifics change. As a result, weights often need to be recalibrated and risk being fragile. Worse yet, the weight factors which encapsulate the relative importance of the various loss components, being folded into the weight factors used to make the loss components and their variations commensurate, can have unpredictable side effects.

In contrast, turning loss components into ranks automatically normalizes their range of values and, to some extent, their variations. Multiplying ranks by weights is then more likely to lead the optimizer as intended: Using the maximum rank across all loss components, a small rank multiplier for a loss component means that it will only be "active" if the value of that loss is atypically bad; Conversely, a large rank multiplier means that the corresponding loss component will often determine the maximum rank, unless the loss component often reaches its very best value within the reference population (the very best

value of a loss component gets the rank 0). In short, the max-rank loss, weighted or not, has built-in loss normalization.

The second advantage of using max-rank scalarization is that it leads to good compromises between conflicting objectives. In other words, it tends to point the optimizer toward regions within hyperparameter space where no loss component attains an atypically bad value. The reason (ignoring weights for the sake of discussion) is that very low ranks for loss components are penalized if they come at the expense of high ranks for other loss components. Thus, the max-rank scalarization effectively communicates to the solver that an hyperparameter setting for which any of the loss components is atypically bad is itself undesirable.

The main disadvantage of using the max-rank loss is that its alignment with good results depends on the reference population. All trials sampled so far within Search Space Reduction are used to compute the corresponding max-ranks. Similarly, all trials visited so far by CMA-ES, including the initial centroid, are used to compute the corresponding max-ranks.

We have found that the max-rank loss effectively guides the proposed optimizer toward better vector losses. For example, optimizing the synthetic ISP's SSIM at both low and high gain, high gain SSIM is significantly better when scalarizing high and low gain losses with the max-rank loss (0.739) than with weighted convex combination (0.716), even though low gain SSIMs are equivalent. (Experiments performed with the same initial hyperparameter setting, the same number of trials, and loss weights set to 1.)

2.2.2 Definition

Consider a set of nonnegative real weights $\{w_j\}_{l=1}^{\mathrm{L}}$. Also consider a collection of N trials $\{\boldsymbol{\Theta}_k\}_{k=1}^{\mathrm{N}} \subset \mathbb{R}_{[0,1]}^{\mathrm{P}}$ with corresponding multi-objective losses $\{\boldsymbol{\mathcal{L}}_k = (\mathcal{L}_1(\mathbf{s}(\boldsymbol{\Theta}_k)), \ldots, \mathcal{L}_{\mathrm{L}}(\mathbf{s}(\boldsymbol{\Theta}_k)))\}_{k=1}^{\mathrm{N}} \subset \mathbb{R}^{\mathrm{L}}$.

The set $\{\boldsymbol{\Theta}_k\}_{k=1}^{\mathrm{N}}$ of (stochastically generated) trials over which the max-rank loss is computed could be all the trials since CMA-ES initialization, initial centroid $\boldsymbol{\Theta}^{(0)}$ included; or all the trials computed so far as is the case at the end of every sampling round of Search Space Reduction.

The set of nonnegative weights $\{w_l\}_{l=1}^{\mathrm{L}}$, the *loss weights*, could be based on statistical analysis (this is the case at the end of each round of Search Space Reduction sampling when used to compute the reduced hyperparameter search intervals); or provided by the user to prioritize some loss components (at the end of Search Space Reduction to compute the "best so far" hyperparameter setting used to initialize CMA-ES, up to the end of the last generation), as is or evolved from these user-supplied values (from the second generation of CMA-ES on).

In any case, once chosen the set of trials ("reference population") and the loss weights, the *max-rank loss* is computed as follows. First, convert each loss component to a rank. Specifically, one loss component $\mathcal{L}_{\mathrm{l}}$ at a time, sort the loss values $\{\mathcal{L}_{\mathrm{l}}(\mathbf{s}(\boldsymbol{\Theta}_k))\}_{k=1}^{\mathrm{N}}$ in non-decreasing order, that is, from best to worse. $\mathcal{R}_l(\boldsymbol{\Theta}_k)$ is then the rank of the kth trial's loss component value within the sorted list, the very lowest rank being set to 0 and ties resolved by taking the smallest rank among the alternatives so that, for example, if all trials get the exact same value for the loss component $\mathcal{L}_l$, all the corresponding ranks equal 0. In summary,

$$\mathcal{R}_l(\boldsymbol{\Theta}_k) = \mathrm{rank}_{\{\mathcal{L}_l(\mathbf{s}(\boldsymbol{\Theta}_1)),\ldots,\mathcal{L}_l(\mathbf{s}(\boldsymbol{\Theta}_{\mathrm{N}}))\}}(\mathcal{L}_l(\mathbf{s}(\boldsymbol{\Theta}_k)).$$

Once the loss components $\mathcal{L}_l$ are converted to ranks $\mathcal{R}_l$, the weighted max-rank loss $\mathcal{M}$ associated with a trial is simply

$$\mathcal{M}(\mathbf{s}(\boldsymbol{\Theta}_k)) = \max_{l \in \{1,\ldots,\mathrm{L}\}} w_l \cdot \mathcal{R}_l(\boldsymbol{\Theta}_k).$$

Each with their own set of trials and weights, the following computations all use the max-rank loss.

2.3. Search Space Reduction

Search Space Reduction is dual-purpose. Firstly, Search Space Reduction smoothly remaps the (relaxed to continuous values and normalized) hyperparameter search hypercube $\mathbb{R}_{[0,1]}^{\mathrm{P}}$ to itself in such a way that, for each hyperparameter, that is, for each dimension, an hyperparameter range correlated with better loss values takes up a larger portion of the unit interval. Secondly, Search Space Reduction provides an improved best so far initial search point to the final optimizer, namely CMA-ES.

Following Search Space Reduction, the entire original search space is still reachable. No hyperparameter value is excluded as would be the case with a *hard box* approach. A hard box approach, which rigidly narrows the search range of each parameter, prevents further optimization from visiting an hyperparameter value that falls outside of the interval estimated to contain best values. Consequently, hard boxes prevent further optimization from recovering from an imperfect Search Space Reduction. Instead, the proposed *soft box* approach biases the search instead of restricting it.

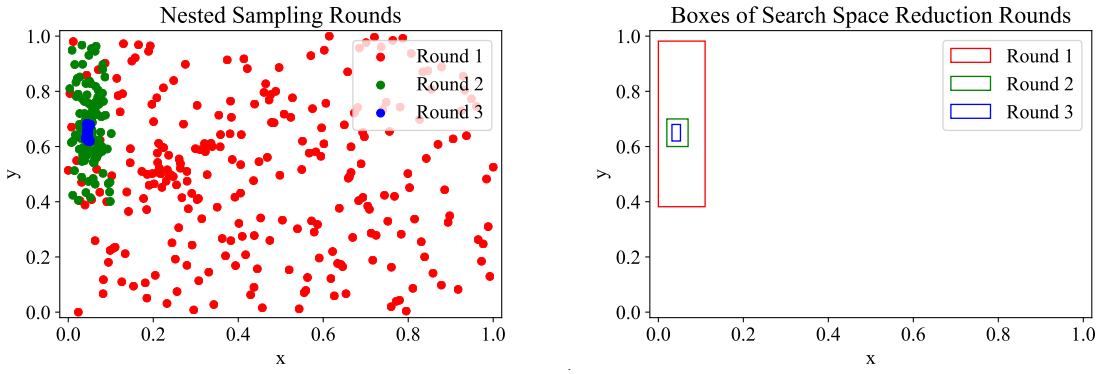

Figure 1: Search Space Reduction example with two hyperparameters (x and y) and three rounds.

We perform three essentially identical Search Space Reduction rounds (Fig. 1). Each round has three steps. First, sample the current hard box, initially the entire hyperparameter space $\mathbb{R}^{\mathrm{P}}_{[0,1]}$ but a smaller hyperrectangle in later rounds. Second, perform statistical analysis over all known trials (samples from the latest sampling round as well as earlier ones) to determine, separately for each hyperparameter, the subinterval of $\mathbb{R}_{[0,1]}$ most likely to contain optimal values, as well as the most likely optimal hyperparameter value. Third, assemble the intervals into a hard hyperrectangular box within which the next Search Space Reduction sampling round, if any, is restricted. Following the last Search Space Reduction round, remap $\mathbb{R}^{\mathrm{P}}_{[0,1]}$ so that the final hard box occupies a larger portion of search space volume. This remapped hypercube search domain, along with a (remapped) best so far extracted from the known trials, is then used by CMA-ES. Additional details follow.

2.3.1 Approximately Latin Hypercube Sampling

Entering each round of Search Space Reduction which uses Gaussian sampling (they all perform some by default), a centroid $\hat{\boldsymbol{\Theta}}$ is chosen. In the first round, $\hat{\boldsymbol{\Theta}}$ is taken to be the default ISP hyperparameter setting. In later rounds, $\hat{\boldsymbol{\Theta}}$ is assembled from hyperparameter values found by statistical analysis; this assembled hyperparameter setting is consequently not a best so far since its loss is not known until the next round of sampling. That is, the centroid is invariably included in the sampling of the next round, bypassing random generation for one trial.

Hard box sampling is performed as follows: Trials are independent. The proportion of the random trials with hyperparameter values sampled from a Gaussian distribution is initially half (by default: increase when the vendor default is a good fit for the end use; reduce otherwise). This proportion increases from one round to the next. The remaining trials are drawn from a uniform distribution. Coordinates are drawn independently, with a fixed standard deviation (scaled by the width of the reduced search interval in later rounds) in the case of Gaussian sampling. If the generated hyperparameter coordinate falls outside of the corresponding unit interval (hard box, in later rounds), it is repeatedly reflected about the nearest boundary until it falls within.

Because the probability of repeating the same hyperparameter coordinate value is 0 in continuous parameter space (ignoring the limitations of machine floating point and random generators, since the set of hyperparameter settings with one fixed coordinate has measure 0), this procedure yields a sampling of the hypercube which is approximately Latin. Note however that since the ISP uses discrete values, hyperparameter values passed to the ISP may repeat. For example, in the first round, each of the two branches of a binary toggle is sampled half the time by the uniform distribution component of the random sampling. That is: In "native ISP register space", search space sampling is generally *not* Latin. Approximate Latinness only holds in floating point.

2.3.2 Estimated Narrowed Search Interval

At the conclusion of each round of sampling, the newly acquired trial data is aggregated with trial data acquired earlier and statistical analysis is performed on the entire lot. Let N be the total number of trials thus aggregated.

The goal of the interval narrowing procedure described in this subsection is to find a small search box within which the CMA-ES-based search will be biased to stay. This smaller search box is obtained by "discarding" large regions of the full (normalized) hyperparameter space. "Discarded" regions are estimated to be unlikely to contain "good" trials, "good"

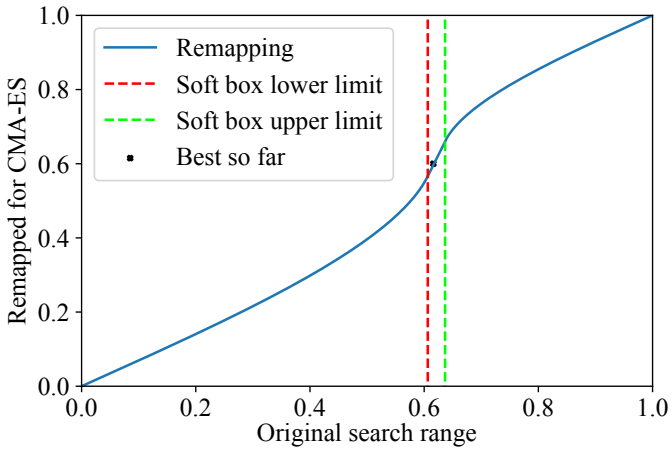

Figure 2: Search Space Reduction soft box remapping for the hyperparameter y of Fig. 1.

trials being those for which no loss component "fails" significantly as assessed by its rank among all known trials. Such "unbalanced" trials, for with at least one loss component has an atypically bad value (weights taken into account), are generally not interesting from the point of view of ISP optimization, and consequently the solver avoids them.

We speak of a "soft box" approach because in the end we do not actually discard "bad" hyperparameter space regions but instead reduce their effective volume, as "seen" by the CMA-ES component of the optimizer, within the full (normalized) search space $\mathbb{R}^{\mathrm{P}}_{[0,1]}$. In a "hard box" approach, the optimizer cannot leave narrowed search ranges; in a "soft box" one, the optimizer can, but it is "discouraged" to do so. This is in fact accomplished by increasing the volume of the "good" hyperbox within $\mathbb{R}^{\mathrm{P}}_{[0,1]}$ by remapping each coordinate (see Fig. 2).

Specifically, for each of the P ISP hyperparameters Θ_p, a narrowed search interval $\left[\mathrm{low}_p, \mathrm{high}_p\right] \subset [0,1]$, estimated to contain likely better values of Θ_p, is produced. $[\mathrm{low}_p, \mathrm{high}_p]$ is basically a one-dimensional convex hull: It is the smallest interval containing all the Θ_p-values of the best performing trials according to a weighted max-rank loss specific to that hyperparameter. The rank of each loss component is weighted in such a way that the loss components most impacted by Θ_p have larger weights. These weights, and consequently, the best performing trials, are computed independently for each hyperparameter. The "best" hyperparameter values assembled into the centroid used by the next round's Gaussian sampling are also computed coordinate by coordinate; they are not taken from one single trial. By construction, they automatically fall within the corresponding "hard box". Details are found in Algorithm 1. (The function $\mathrm{computeWeight}$ referenced in Algorithm 1 is detailed in Algorithm 2.)

The real-valued weights are in $\mathbb{R}_{[0,1]}$. For each hyperparameter Θ_p, and each loss component $\mathcal{L}_l$, the weight $w_{p,l}$ quantifies the impact of the hyperparameter Θ_p on the loss $\mathcal{L}_l$. A value of 0 for the weight $w_{p,l}$ is understood to mean that the value of Θ_p has no impact on loss $\mathcal{L}_l$; a value of 1 corresponds to a strong impact.

Specifically, weights are set to the value $1 - p_{\mathrm{value}}$, where p_{value} is the p-value of the two-sample Kolgomorov-Smirnov independence test computed by setting the number of observations to a small value (n_{ref}). The computation of the narrowed search interval and best value is detailed in Algorithm 2, where the function K is the large sample Kolgomorov cumulative distribution function described in [10], and cdf stands for "cumulative distribution function".

When performing single objective optimization, the algorithm reduces to finding the smallest box containing the best $\lfloor p_{\mathrm{good}} \cdot \mathrm{N} \rfloor$ trials, where p_{good} stands for the proportion of best trials to consider. best_p is then the pth coordinate of the best trial, namely the trial which minimizes the loss among those within the reference population.

Given that search intervals for different hyperparameters are reduced independently, one may wonder whether this procedure could discard regions of the hyperparameter space where synergies between several hyperparameters give good results. For instance we may have a low loss when both hyperparameters Θ_1 and Θ_2 are close to 1 but a high loss when only one of them is close to 1. Would the values of those hyperparameters be (softly) bounded away from 1 by the narrowing procedure? This is not the case (provided the synergy is "sampled"): Regions of hyperparameter space that are good because of synergies between several hyperparameters are not discarded because, even though we project hyperparameter vectors onto one dimension and separably reduce intervals, we keep all good trials. In particular, we do not base the narrowing on some local statistics, like local averages, as those could lose the information contained in a good trial which sampled an hyperparameter setting close enough to the synergy.

Algorithm 1 Reduced search interval computation for hyperparameter Θ_p.

Require: $p \in \{1, \ldots, \mathrm{P}\}, (p_{\mathrm{good}}, p_L, p_U) \in \mathbb{R}^3_{(0,1)}, \mathrm{N} \in \mathbb{N}^*, \{\boldsymbol{\Theta}_k\}_{k=1}^{\mathrm{N}} \subset \mathbb{R}^{\mathrm{P}}_{[0,1]}$

for $k = 1$ to N **do**
  for $l = 1$ to L **do**
    $r_{k,l} \leftarrow \mathrm{rank}_{\{\mathcal{L}_l(\mathbf{s}(\boldsymbol{\Theta}_1)),\ldots,\mathcal{L}_l(\mathbf{s}(\boldsymbol{\Theta}_{\mathrm{N}}))\}} \left(\mathcal{L}_l\left(\mathbf{s}\left(\boldsymbol{\Theta}_k\right)\right)\right)$
  end for
end for
for $l = 1$ to L **do**
  $L_l \leftarrow \{k \in \mathbb{N}^* \,|\, k \leqslant \mathrm{N}, r_{k,l} \leqslant p_L \cdot \mathrm{N}\}$
  $U_l \leftarrow \{k \in \mathbb{N}^* \,|\, k \leqslant \mathrm{N}, r_{k,l} > p_U \cdot \mathrm{N}\}$
end for
$\mathcal{P} \leftarrow (\Theta_{k,p})_{k=1,\ldots,\mathrm{N}}$
for $l = 1$ to L **do**
  $w_l \leftarrow \mathrm{computeWeight}(\mathcal{P}, L_l, U_l)$ (Algorithm 2)
end for
$\mathcal{M} \leftarrow \left(\max_{l \in \{1,\ldots,\mathrm{L}\}} w_l \cdot r_{k,l}\right)_{k=1,\ldots,\mathrm{N}}$
$\mathcal{S} \leftarrow \mathrm{argsort}(\mathcal{M})$
$\mathcal{B} \leftarrow (\mathcal{S}_k)_{k=1,\ldots,\lfloor p_{\mathrm{good}} \cdot \mathrm{N} \rfloor}$
$\mathrm{low} \leftarrow \min_{k \in \mathcal{B}} \Theta_{k,p}$
$\mathrm{high} \leftarrow \max_{k \in \mathcal{B}} \Theta_{k,p}$
$\mathrm{best} \leftarrow \Theta_{\mathcal{S}_1,p}$
return $(\mathrm{low}, \mathrm{high}, \mathrm{best})$

Algorithm 2 Function computeWeight:
Loss component weight computation based on estimated dependence.

Require: $\mathcal{P} = (p_1, \ldots, p_{\mathrm{N}}) \in \mathbb{R}^{\mathrm{N}}_{[0,1]}, L \subset \{1, \ldots, \mathrm{N}\}, U \subset \{1, \ldots, \mathrm{N}\}, n_{\mathrm{ref}} \in \mathbb{N}^*$

$\mathcal{P}_L \leftarrow \{p_i \,|\, i \in L\}$
$\mathcal{P}_U \leftarrow \{p_i \,|\, i \in U\}$
$d = \sqrt{0.5 \cdot n_{\mathrm{ref}}} \sup |\,\mathrm{cdf}(\mathcal{P}_L) - \mathrm{cdf}(\mathcal{P}_U)|$
$w = K(d)$
return w

It should also be noted that even though the narrowing procedure conservatively keeps good trials, it still achieves a considerable reduction of the volume in hyperparameter space. Typically, one round of the narrowing procedure roughly halves the interval (typical length ratios range from 0.4 to 0.7), which leads to a large volume reduction when there are many hyperparameters. For an ISP with 25 hyperparameters, three rounds of the proposed Search Space Reduction have been found to reduce the volume of the search hyperbox by a factor of 10^{-4} to 10^{-10}.

The use of rank weights tied to p-values results in significantly more aggressive narrowing of the interval in some cases. When trials with both high and low values of some loss component $\mathcal{L}_l$ are spread across the range of a given hyperparameter Θ_p, the unweighted version cannot achieve a significant narrowing of the corresponding interval. For this reason, it is desirable to reduce the impact of this loss component by multiplying the corresponding ranks by a small weight. Such a weight is produced by the function computeWeight (Algorithm 2). Correlating weights with the impact of each hyperparameter on loss components allows significant narrowing while making it unlikely that regions likely to contain "balanced" Pareto points are "discarded". The value of n_{ref} impacts the aggressiveness of the weighting; specifically, a lower value tends to equalize the weights. We use $n_{\mathrm{ref}} = 10$.

2.3.3 Separable Search Hypercube Remapping

Once the final reduced search interval $\left[\mathrm{low}_p, \mathrm{high}_p\right]$ for each of the P hyperparameter coordinates is obtained, the bijective remapping of the unit search hypercube $\mathbb{R}^{\mathrm{P}}_{[0,1]}$ onto itself is constructed as follows. Note that we actually construct the

mapping from remapped space to non-remapped space, since what we really need is to connect hyperparameter values generated by CMA-ES to native ISP hyperparameter values. The construction is separable. First, $[\mathrm{low}_p, \mathrm{high}_p]$ is stretched within $[0, 1]$, growing outward so that the left and right pieces of the unit interval which are outside of it are reduced by the same factor. This factor is chosen so that the length of the "good" interval is doubled (by default). Let $[\mathrm{LOW}_p, \mathrm{HIGH}_p]$ be the stretched interval. The mapping from "CMA-ES space" to "normalized hyperparameter space" is then defined by the unique twice continuously differentiable natural cubic spline with knots at 0, LOW, HIGH and 1 which maps 0 to 0, LOW to low, HIGH to high and 1 to 1. Degenerate cases are stably handled. For example, $[\mathrm{low}_p, \mathrm{high}_p]$ is stretched outward by a small amount if its length is very close to 0. This may happen, for example, if the value of the pth hyperparameter of the "best" trials all correspond to the same ISP register value.

Fig. 2 shows the spline remapping for one hyperparameter.

Internal CMA-ES parameters are then scaled so that the effect is the same as making the remapped softbox a unit hypercube (thus stretching the search domain so that values outside of the softbox are "far").

2.3.4 Max-rank Initialization

At the conclusion of Search Space Reduction, the trial with the lowest value for the the max-rank loss computed over all the trials with known losses, using the rank weights chosen by the user (instead of those computed by Algorithm 2) is chosen as initial hyperparameter setting for CMA-ES. This best so far almost invariably falls within the final Search Space Reduction hard box which is used as a soft box by CMA-ES.

2.4. CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

The last stage of the proposed optimization method is a variant of so-called Active CMA-ES [4]. CMA-ES optimization is driven by the weighted max-rank losses of the trials of each generation (together with a "best so far" updated at every generation). The proposed variant of CMA-ES uses a fixed generation size λ equal to $\frac{4}{3}$ times the number of parameters, rounded up to a multiple of 4. One quarter of the trials of every generation, less than usual, is discarded to compute the centroid used for the Gaussian generation of the hyperparameter values of the trials of the next generation, as well as other CMA-ES internals like the covariance σ, the normalized covariance matrix C and the path variables ps and pc. The number of non-discarded trials is consequently at least equal to the number of hyperparameters being optimized.

The proposed active CMA-ES uses the best so far described in Sec. 2.3.4 as initial centroid, and uses the search domain remapping described in Sec. 2.3.3. Additional modifications are detailed below.

2.4.1 Adaptive Weighted Max-rank Loss

Within each generation, the rank associated with each loss component is multiplied by an optional weight chosen by the user to prioritize some loss components in the computation of the weighted max-rank loss. (These weights are 1 by default.) As mentioned earlier, weights have no impact when performing single objective optimization. The following additional modification, adaptive loss weighting, also has no impact when none of the MOO loss components can attain the "good enough" or "pass" value 0. When optimizing for perceptual IQ, however, "good enough" values for some quality metrics are often known. (For example, they may be based on Just Noticeable Differences (JND).) In that case, the following methodology often leads to converged hyperparameter settings which fall just a bit short of "passing" when multiple losses are in opposition.

The purpose of this novel CMA-ES modification is to focus the optimization on the loss components which are harder to "pass". Here is how adaptive max-rank loss weighting is implemented: The loss weights are fixed within a generation and are updated at its conclusion. Within each generation and for each loss component, the proportion of trials that fail to attain the "good enough" loss value 0 is noted. These proportions, one per loss component, are initialized to 1 and are updated in a Markovian fashion with a time horizon just long enough to prevent oscillations. When computing the weighted max-rank loss with reference population set to the last completed generation, these evolving weights which track, for each loss component, how difficult it has been to attain the "good enough" value 0 in recent generations, are folded into the user-specified loss component weights by multiplication. In any case, a loss component which has reached the value 0 often in recent history has its impact on the max-rank loss diminished.

Adaptive max-rank loss could be used with other optimization methods than CMA-ES, the necessary ingredient being that the method have a notion of "generation" over which the weights are fixed and between which they change.

2.4.2 Active Centroid Weights

The centroid used to generate the next generation's trials is a weighted average of the coordinates of the non-discarded trials of the current generation. The proposed CMA-ES variant uses so-called "active" centroid weights, meaning that some of them are negative [4]. (Early versions of CMA-ES only used positive weights.) Unlike the weights discussed so far, these geometrical weights multiply hyperparameter coordinate values instead of losses.

A novel weight construction is used by the proposed CMA-ES variant. Taking note that the number of non-discarded trials is automatically a multiple of three (because we retain three quarter of a number of trials which is a multiple of four), they are constructed as follows, before being normalized by their sum, as befits a weighted averaging computation. Based on the weighted max-rank loss, rank the non-discarded trials of the generation from best to worse (the discarded trials would have worse ranks anyway). Let r be the rank associated with a trial, and let $3n$ ($n \in \mathbb{N}^*$) be the number of non-discarded trials, within the generation. The rank r consequently ranges from 0 (best) to $3n-1$ (worst non-discarded). Resolve ties by averaging; For example, if the four best trials have the exact same weighted max-rank loss (and the fifth does not), the four best trials all get $r = (0+1+2+3)/4 = 1.5$. Then, the pre-normalization centroid weight ω used with the hyperparameter coordinates of this trial is set to $2n - r - 1/2$. Ignoring ties, the negative weights are canceled in a pairwise manner by the smallest positive ones. Thus, one can interpret the contribution of the second and third quartiles (the fourth quartile being discarded) to the centroid computation as vector differences, the vector pointing from poor values to better ones. Following normalization, the weights are $\frac{4n-2r-1}{3n^2}$.

2.4.3 Greedy "Best So Far" Centroid Substitution

As seen above, as in vanilla CMA-ES, centroids are computed by weighted averaging of the trials of the last generation. Under some conditions however the proposed method branches out and uses a different centroid.

At all times, a "best so far" is maintained and updated. Using the current values of the adaptive weights, the max-rank of the best so far is compared to the trials of the generation. If the generation contains trials that have a better weighted max-rank loss, the best one (with ties resolved randomly) replaces the "best so far" as well as the weighted average centroid. Internal CMA-ES parameters are computed before this substitution. This guides the convergence more aggressively toward balanced hyperparameter settings within the Pareto front.

When performing single objective optimization, this new "best so far" is actually a global best so far. (Note that in order to foster exploration and the discovery of stable solutions, the above branch switches to a new best so far when there is a global tie.) When performing MOO, however, it may not be. The reason is that an earlier trial may happen to have a better max-rank *with respect to the new weights*. Such "historical best so far's" are ignored. (They are, of course, included in the final Pareto sort.)

We performed an ablation test to validate the effectiveness of the proposed greedy variant of vanilla CMA-ES: The domain of the 10-dimensional Rosenbrock function was discretized. Using the same number of trials (set so that the normalized standard deviation is smaller than 5) and initial starting point, we obtained a 10.4 loss with the proposed greedy centroid substitution, and a 11.3 loss turning it off.

2.4.4 Boundary Handling by Reflection

As is standard with CMA-ES, the trials of the next generation are generated by random Gaussian generation. (Normalized) hyperparameters that fall outside of the unit hypercube $\mathbb{R}^{\mathrm{P}}_{[0,1]}$ are mapped back into it by repeatedly reflecting each coordinate about the closest unit interval endpoint until it lands inside [3]. This is efficiently implemented with a combination of the modulo 2 and absolute value operations, and is done first thing, before CMA-ES' internal statistics are computed. Because some of the active centroid weights are negative, centroids may also fall outside of the unit hypercube; they are likewise mapped back into it before use.

2.4.5 Robustness with Respect to Discrete Nature of ISP Hyperparameters

CMA-ES uses an estimated covariance matrix—the product of a scaling factor generally called "σ" and of an approximately normalized symmetric positive definite matrix generally called "C"—to change the scale to localise or globalise the exploration and to stretch the Gaussian sampling in directions estimated to be desirable (at the current scale), and shrink it in directions estimated to be less desirable. This estimated covariance matrix is updated every generation. Convergence and random variations may cause σ and/or some eigenvalues of C to get so small that most of the trials of a generation have

hyperparameters with the exact same value once converted to native, discrete, ISP register values. This leads to insufficient exploration of the configuration space, and affects the accuracy of the internal statistics that guide optimization, for example, the estimated covariance matrix itself.

This is addressed by the proposed CMA-ES variant as follows. In order to prevent an ISP hyperparameter from getting stuck at a fixed discrete value, Gaussian noise is added to each drawn hyperparameter value with a standard deviation chosen so that the proportion of trials without a bit change is approximately the proportion of trials with positive centroid weights.

Specifically, after a trial's hyperparameter coordinates have been generated, and before they are reflected back, if necessary, into the unit hypercube, random Gaussian noise is added to each coordinate. This Gaussian noise is separate from, in fact independent of, the (vanilla) CMA-ES σ and C used to draw the trials. That is: Unlike vanilla CMA-ES trial generation which varies along with the covariance matrix, the distribution of the added noise is constant through optimization.

The distribution of the Gaussian noise added to an hyperparameter coordinate in $\mathbb{R}_{[0,1]}$ depends on two quantities: First, the number B of different values found in the range of values of the corresponding ISP register; Second, the total number P of hyperparameters being optimized. For example, if optimizing an 8-bit hyperparameter over its full range, B would be 256; when optimizing over only part of its range (because some values are known to give suboptimal values), B would be the actual number of discrete values within the narrowed range.

Recall that native, discrete, integer values of the hyperparameter $\{\text{lowest value}, \ldots, \text{lowest value}+\mathrm{B}-1\}$ are relaxed to the unit interval $\mathbb{R}_{[0,1]}$. Also, let p be the proportion of the trials of each generation which do not have a positive centroid weight ($\mathrm{p} = 1/2$ with the proposed variant of CMA-ES). The standard deviation of the added Gaussian noise is chosen so that, on the one hand, the probability of triggering a bit change is the same for all hyperparameters, and on the other hand, so that the proportion of trials with one or more changed bit as a result of adding noise is at least p (on average). Ignoring boundary issues, the simplifying assumption that each relaxed hyperparameter is at the center of the basin of attraction of an integer ISP register value (this is the most stable position) leads to a value of the standard deviation equal to

$$\sigma_{\text{noise}} = \frac{1}{2\,(\mathrm{B}-1)\,\mathrm{cdf}^{-1}\!\left(\frac{1}{2}+\frac{1}{2}\left(1-\mathrm{p}\right)^{1/\mathrm{P}}\right)},$$

where cdf^{-1} is the inverse of the Cumulative Density Function of the Gaussian distribution with unit standard deviation. (Note that if the relaxed hyperparameter value is at the edge of the basin of attraction, the probability of triggering a bit change is at least one half (ignoring boundary issues).) Adding Gaussian noise with standard deviation σ_{noise}, the probability that one hyperparameter have a bit change in ISP native space is consequently $1-(1-\mathrm{p})^{1/\mathrm{P}}$. This implies that the probability that *none* of the coordinates of a trial's hyperparameters have a bit change is

$$\left(\left(1-\left(1-(1-\mathrm{p})^{1/\mathrm{P}}\right)\right)\right)^{\mathrm{P}} = 1-\mathrm{p}$$

as prescribed.

We performed an ablation test to evaluate the impact of this CMA-ES enhancement by optimizing 3-, 4- and 8-bit OnSemi AP0202AT ISP hyperparameters (one of them restricted to a 7-bit range outside of which the parameter has little effect). (See Sec. 1.2.) The experiments were consequently performed in a context in which the correspondence between discrete values and the unit interval is not contrived. Only one evaluation metric, namely PSNR, was used, to keep MOO issues out of the way; the experiment was otherwise similar to OnSemi experiment described in Sec. 6.3 of the main document. With the same initial hyperparameter setting and total number of trials, PSNR reached 11.6 dB with vanilla CMA-ES trial generation, and 12.0 dB with Gaussian noise added as described above.

2.4.6 Monte Carlo Simulation of Deviations from Randomness

Vanilla CMA-ES estimates deviations from randomness using a χ_N estimate based on the assumption that the hyperparameters' distribution is Gaussian. This assumption, however, is both explicitly and implicitly violated by the proposed optimizer. In the proposed variant of CMA-ES, the usual estimates are replaced by a tracking Monte Carlo simulation based on random rank generation.

We performed an ablation test to validate the effectiveness of the proposed Monte Carlo statistics generation: The domain of the 10-dimensional Rosenbrock function was discretized. Using the same number of trials (set so that the normalized standard deviation is smaller than 5) and initial starting point, we obtained a 10.4 loss with Monte Carlo simulation, and a 14.3 loss using standard CMA-ES estimates.

2.4.7 Postmortem Pareto Sort

At the conclusion of CMA-ES, all (known) trials, sampled during Search Space Reduction and CMA-ES, intial points included, are aggregated, and a Pareto sort performed over the entire set. The Pareto point that appeared last is generally taken to be the final, optimal, hyperparameter setting. The Pareto front itself is generally informative in and by itself.

3. Comparisons with Other Solvers

We benchmarked the proposed ISP optimization algorithm against three other state-of-the-art hyperparameter optimization methods, namely HyperOpt [5], BayesOpt [20] and vanilla CMA-ES [16], in two domain applications: perceptual image quality and automotive 2D object detection.

First, we present comparative results for the synthetic ISP. The synthetic ISP was optimized for the perceptual image quality loss $\mathcal{L}_{\text{PERC}} = 5\hat{\mathcal{L}}_{\text{PERC}} + \ell_1$ described in Sec. 6.3 of the main document. In all the experiments, we set the maximum number of function evaluations to 10000, and the initial point was the same. Fig. 3 shows convergence profiles for the single objective optimization experiments. Only vanilla CMA-ES and the proposed method converged within 10000 function evaluations. The proposed method converged within 5000 function evaluations; vanilla CMA-ES within 7000. Table 4 shows the values of the perceptual image quality loss $\mathcal{L}_{\text{PERC}}$ for the final result of each of the methods. Additional perceptual image quality metrics, namely SSIM, PSNR and CIELABΔE, are also shown. The proposed optimization method outperforms the other optimization methods with respect to all evaluation metrics.

With the synthetic ISP, we also performed a comparison of MOO optimization of automotive 2D object detection. Two evaluation metrics were used: mAP and mAR. With HyperOpt, BayesOpt and vanilla CMA-ES, we used the average of mAP and mAR as objective function; this is equivalent to scalarizing the losses with equal weight convex combination. With the proposed optimization method, we used the proposed adaptive weighted max-rank scalarization (with "user weights" equal to 1). As shown in Table 5, the proposed method resulted in significantly better mAP *and* mAR scores than the alternatives.

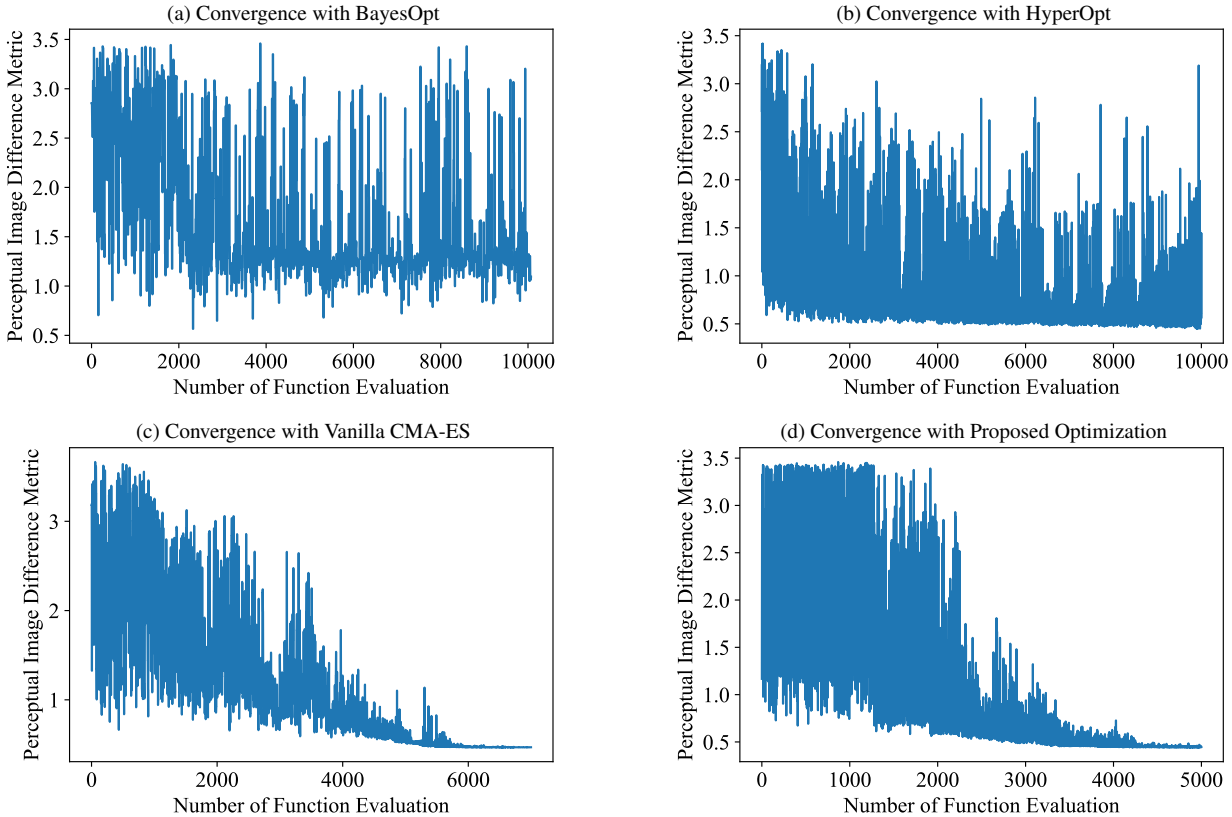

Figure 3: Convergence profiles for the optimization of the synthetic ISP for the perceptual loss $5\hat{\mathcal{L}}_{\text{PERC}} + \ell_1$ with (a) BayesOpt [20], (b) HyperOpt [5], (c) Vanilla CMA-ES[16], and (d) the proposed optimization method.

Table 4: Perceptual image quality evaluation metrics measured on the final results of synthetic ISP optimization.

	BayesOpt [20]	HyperOpt [5]	Vanilla CMA-ES [16]	Proposed Optimization
$5\hat{\mathcal{L}}_{\text{PERC}} + \ell_1$	0.566	0.448	0.468	**0.442**
SSIM	0.656	0.680	0.677	**0.697**
PSNR	19.25	19.41	19.55	**19.79**
CIELABΔE	10.10	9.85	9.41	**8.31**

Table 5: Effect of Optimization Method for ISP Hyperparameter Optimization.

	BayesOpt [20]	HyperOpt [5]	Vanilla CMA-ES [16]	Proposed Optimization
mAP	0.60	0.68	0.69	**0.75**
mAR	0.57	0.64	0.66	**0.71**

4. Additional Assessment

4.1. ISP Optimization for Image Understanding

To validate the proposed approach with the ARM MALI-C71 ISP, we collected 500 RAW images with the SONY IMX249 sensor for automotive object detection. Of the 500, 60 were used for ISP optimization and the rest for evaluation. Captures were made in various conditions including snowfall, rainfall, high sunlight and tunnels, in daytime and nightime. The RAW images were carefully annotated to include all pedestrians and cars appearing within the RAW with a height and width of more than 10 pixels. Prior to expert tuning and optimization, as explained in Sec. 1.1, the ISP's white balance gains were calibrated, and they were kept fixed thereafter.

Comparison Figures 4, 5, 6, 7, 8 show images processed with the default vendor ISP hyperparameters; with ISP hyper-

(a) Detection with Default ISP Hyperparameters

(b) Expert-tuned ISP Hyperparameters for Perceptual Image Quality

(c) End-to-end ISP Optimization with ISP Approximation from Tseng *et al.* [23]

(d) Hardware-in-the-loop ISP End-to-end Optimization for Object Detection

Figure 4: ARM MALI-C71 ISP automotive 2D object detection example in a tunnel. With the ISP optimized using the proposed hardware-in-the loop approach, noise is visible; however, the object detector manages to detect smaller objects with this ISP hyperparameter setting than with hyperparameters obtained with alternative methods, presumably because smoother images have less detail.

(a) Detection with Default ISP Hyperparameters

(b) Expert-tuned ISP Hyperparameters for Perceptual Image Quality

(c) End-to-end ISP Optimization with ISP Approximation from Tseng *et al*. [23]

(d) Hardware-in-the-loop ISP End-to-end Optimization for Object Detection

Figure 5: ARM MALI-C71 ISP automotive 2D object detection example in strong sunlight. The image obtained with hyperparameters optimized with the proposed hardware-in-the-loop method has more saturated pixels than those obtained with alternatives, a setting which most experts would find undesirable for human viewing; however, more cars are correctly detected.

(a) Detection with Default ISP Hyperparameters

(b) Expert-tuned ISP Hyperparameters for Perceptual Image Quality

(c) End-to-end ISP Optimization with ISP Approximation from Tseng *et al*. [23]

(d) Hardware-in-the-loop ISP End-to-end Optimization for Object Detection

Figure 6: ARM MALI-C71 ISP for 2D automotive object detection example in snowfall condition.

(a) Detection with Default ISP Hyperparameters

(b) Expert-tuned ISP Hyperparameters for Perceptual Image Quality

(c) End-to-end ISP Optimization with ISP Approximation from Tseng *et al.* [23]

(d) Hardware-in-the-loop ISP End-to-end Optimization for Object Detection

Figure 7: ARM MALI-C71 ISP for 2D automotive object detection example in rainy condition.

(a) Detection with Default ISP Hyperparameters

(b) Expert-tuned ISP Hyperparameters for Perceptual Image Quality

(c) End-to-end ISP Optimization with ISP Approximation from Tseng *et al.* [23]

(d) Hardware-in-the-loop ISP End-to-end Optimization for Object Detection

Figure 8: ARM MALI-C71 ISP for 2D automotive object detection example in night snowfall conditions.

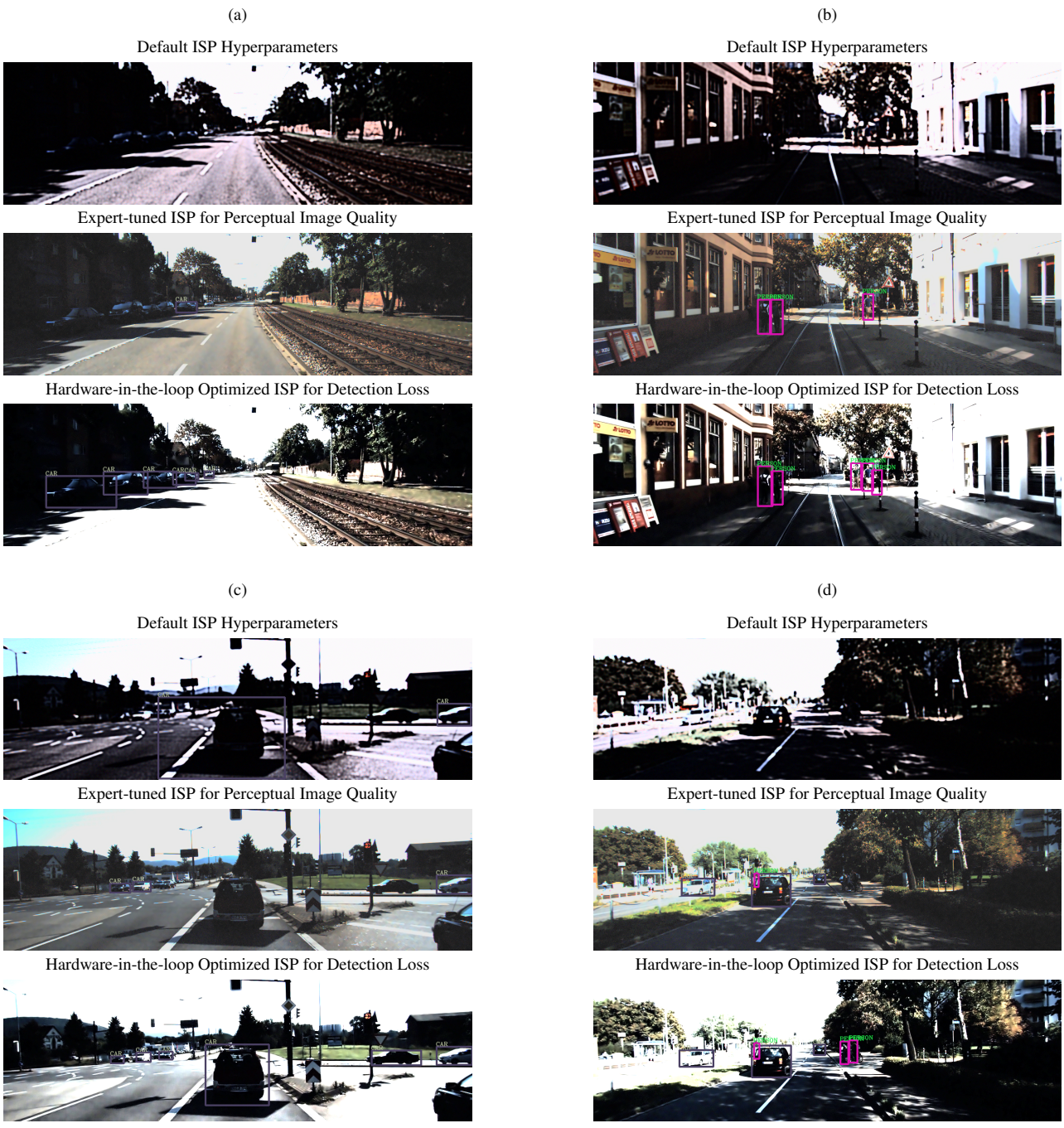

Figure 9: Additional image understanding results for automotive object detection on KITTI using [22]: default ISP hyperparameters; hyperparameters expert-tuned for perceptual image quality; and task-specific-optimized ISP hyperparameters, respectively. (a)-(b) correspond to four different images from the evaluation dataset.

parameters expert-tuned for perceptual image quality; with ISP hyperparameters optimized for object detection loss using the differentiable ISP approximation [23]; and with ISP hyperparameters optimized for the same loss with the proposed hardware-in-the-loop end-to-end method.

The ISP optimized with the proposed hardware-in-the-loop method denoises less; this is clearly visible in the tunnel image (Fig. 4). However, the loss of detail associated with the other methods' stronger denoising caused the downstream automotive

object detector to miss smaller cars detected with the proposed approach.

Fig. 5 illustrates automotive object detection in strong sunlight. The proposed method resulted in inaccurate colors and a larger number of saturated pixels. This however minimally impacted detection performance: The ISP optimized using the proposed method still outperformed the alternatives. Note however that if color accuracy had been key for the downstream application, as is the case for traffic-light detection, this hyperparameter setting, produced by optimizing with respect to an end-to-end loss for which color is relatively unimportant, would most likely not be optimal. Optimization results depend strongly on the loss, which itself depends on the downstream task (pedestrians and cars vs. traffic lights, for example).

ISP hyperparameters obtained with the proposed hardware-in-the-loop method also outperformed the alternatives at night (Fig. 8), in snowfall (Fig. 6, 8), and in rainfall (Fig. 7).

We also show additional visual results for the synthetic ISP image understanding experiments (Sec. 6.1 and 6.2 of the main document) in Fig. 9 and Fig. 10.

4.2. ISP Optimization for Human Viewing

We also validated the proposed hardware-in-the-loop approach by optimizing two hardware ISPs, namely the ARM Mali-C71 and the OnSemi AP0202ATISP, for perceptual (human viewing) image quality. Details omitted in the main document are found below.

First, the camera was aligned and focused on a LCD with the help of a chart filled with spoke targets. We positioned the LCD so that each of its pixels was smaller than the sensor pixel in the output image. In our experiments a single iMac 5K LCD was sufficient to cover most of each camera's field of view. Multiple displays are used with cameras with a larger field of view.

Like in [23], we measure the correspondence between sensor and LCD pixels using Gray code calibration [14]. We then resample the displayed target using the inverse of the calibrated light-transport matrix, resulting in an *aligned reference* registered to the ISP output. Downsampling was performed as a weighted average applied to the nearest neighbors within the displayed chart as determined by a kd-tree computation, with weights that vary inverse linearly with distance. This emulates Elliptical Weighted Averaging with a "cone" kernel.

To compute full reference image difference metrics, we used the random ellipses region of a variant of the Rainbow Chart described in [23], the main differences being that the random ellipses region does not contain a grayscale sub-region and that the linear light average of the random ellipses corresponds to middle gray instead of the average of the patches of an X-Rite ColorChecker. The aligned reference was also tone mapped with a contrast-boosting S-curve and sharpened. The resulting *enhanced (aligned) reference* (Fig. 11) was used as the reference in the loss function. One loss was computed per gain. Specifically, the loss function corresponding to each capture was the value of the image difference between the output image and the enhanced reference quantified with $\mathcal{L}_{\text{PERC}} = 5\hat{\mathcal{L}}_{\text{PERC}} + \ell_1$ within a region of interest that covers the random ellipses.

Some of the hyperparameters of ARM Mali-C71 are gain-specific. They include: thresh 1h, strength 1, thresh 4h, thresh long, sharp alt ld, sharp alt ldu, sharp alt lu and np offset. These parameters were modulated for each gain. All other hyperparameters were shared across the gains. The optimizer evaluated the losses for both the low and high gain captures, and simultaneously updated all ISP parameters, whether shared by all gains, low gain-specific, or high gain-specific. Note that, for each gain, an appropriate combination of LCD brightness and exposure time was found such that there was no flickering and the output was not over or underexposed using the chosen, fixed, manual settings.

4.3. Details of Blockwise ISP Optimization

In the main document, we validated the proposed method against the blockwise ISP optimization method of Nishimura *et al*. [21]. The authors optimize the blocks of an ISP sequentially, one set of hyperparameters, associated with a specific block of the ISP, at a time, which requires detailed knowledge of the operation of each processing block. We used the synthetic ISP.

Evaluation was performed with the eighty-class object detection end-to-end loss described in the main document. Recall that the synthetic ISP has six tunable blocks, see Sec. 1.3. Optimization was initialized with default hyperparameters. Each block was optimized with Algorithm 1 (see the main document) restricted to the hyperparameters specific to the considered block, keeping the hyperparameters of the other blocks fixed. Specifically, when optimizing the n^{th} tunable block of the ISP, the hyperparameters for the earlier blocks $1, ..., n-1$ were fixed to the optimal values found in previous optimization stages, and the hyperparameters of the later blocks $n+1, ..., 6$ were set to ISP default values.

Table 6 shows the mAP and mAR scores after each of the optimization stages performed for the processing blocks listed in Sec. 1.3. As explained above, at each stage, blocks that appear later in the pipeline use default hyperparameter values, and blocks that appear earlier use previously optimized ones. Consequently, the final end-to-end optimized hyperparameters

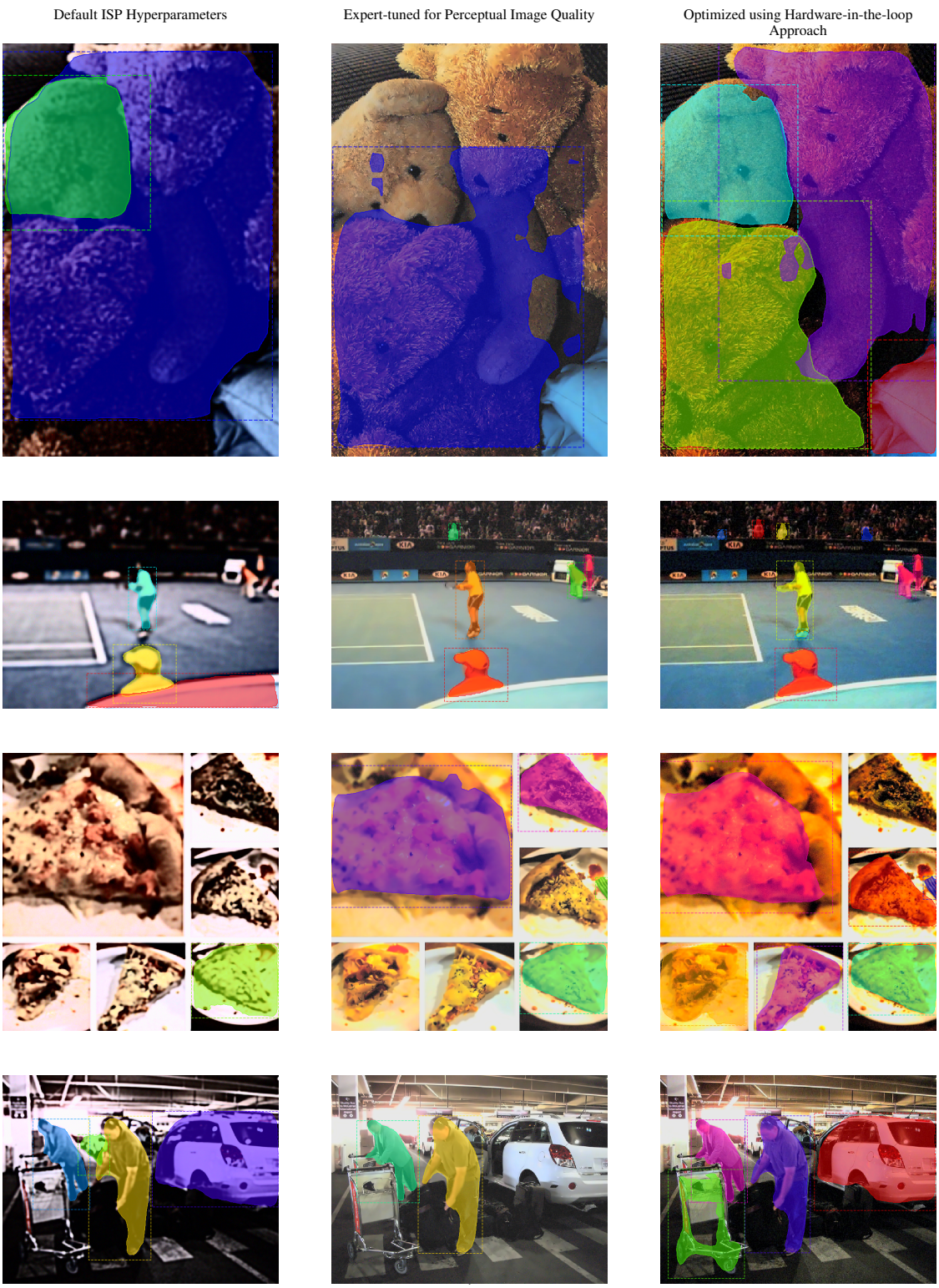

Figure 10: Additional image understanding results for instance segmentation on COCO using [19]: default ISP hyperparameters; hyperparameters expert-tuned for perceptual image quality; and task-specific-optimized ISP hyperparameters, respectively. Each row corresponds to an image from the evaluation dataset.

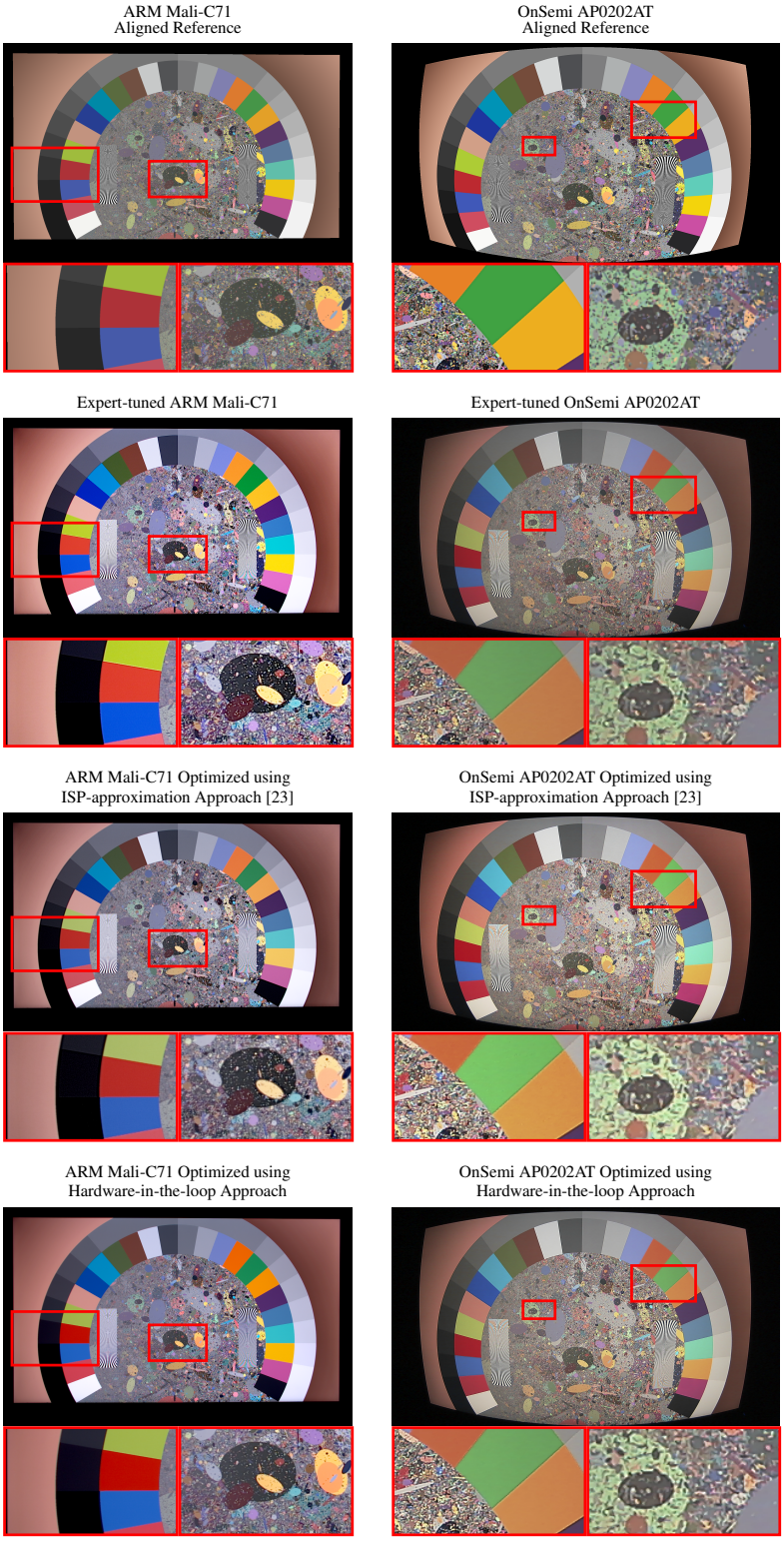

Figure 11: Aligned reference images used in the optimization of ARM Mali-C71 and OnSemi AP0202AT ISPs for perceptual image quality loss alongside the results of processing with hyperparameters obtained by an expert, with the approximation-based optimization method [23], and with the proposed hardware-in-the-loop optimization method.

Table 6: mAP and mAR scores measured over the evaluation set after each stage of the blockwise optimization process for the eight-class object detection loss.

Optimization Stage	default	Luminance Denoiser	Gaussian Denoiser	Chroma Denoiser	Sharpening	Contrast Stretching	Tone Mapping
mAP	0.147	0.160	0.165	0.17	0.168	0.185	0.204
mAR	0.126	0.127	0.145	0.14	0.145	0.158	0.173

are those obtained at the last stage, when the hyperparameters that modulate the tone mapping block of the synthetic ISP are optimized. Scores improve significantly when the 5^{th} and 6^{th} blocks are optimized. This suggests that the contrast stretching and tone mapping blocks of the ISP most impact the object-detection loss. This is not surprising since contrast amplifies image gradients.

The evaluation scores, with the final end-to-end hyperparameter setting obtained with blockwise optimization, did not match that obtained with the proposed joint optimization method: As shown in Table 2 of the main document, both the mAP and mAR are 19% better with the proposed joint optimization. This suggests that ISP blocks do not independently affect end-to-end loss. This also suggests that overall performance cannot be significantly improved by expert tuning if, as suggested in [25], it only touches the hyperparameters that appear to have the most impact on the downstream task (*e.g.* contrast enhancement).

References

[1] AP0202AT high-dynamic range (HDR) image signal processor (ISP). https://www.mouser.com/datasheet/2/308/AP0202AT-D-932936.pdf. Accessed: 2020-03-30. 2

[2] MALI CAMERA C71 image signal processing for automotive. https://www.arm.com/products/silicon-ip-multimedia/image-signal-processor/mali-c71. Accessed: 2020-03-30. 1

[3] Jarosław Arabas, Adam Szczepankiewicz, and Tomasz Wroniak. Experimental comparison of methods to handle boundary constraints in differential evolution. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 411–420, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 10

[4] 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. 3, 9, 10

[5] James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning (ICML)*, 2013. 12, 13

[6] Henryk Blasinski, Joyce Farrell, Trisha Lian, Zhenyi Liu, and Brian Wandell. Optimizing image acquisition systems for autonomous driving. *Electronic Imaging*, 2018(5):1–7, 2018. 2

[7] Michael S. Brown. Understanding the in-camera image processing pipeline for computer vision. *IEEE International Conference on Computer Vision (ICCV) - Tutorial*, 2019. 2

[8] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-local means denoising. *Image Processing On Line*, 1:208–212, 2011. 2

[9] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. Reconfiguring the imaging pipeline for computer vision. In *IEEE International Conference on Computer Vision (ICCV)*, pages 975–984, 2017. 2

[10] James Durbin. *Distribution Theory for Tests Based on the Sample Distribution Function*. SIAM, 1973. 7

[11] Gabriel Eilertsen, Rafał K Mantiuk, and Jonas Unger. A comparative review of tone-mapping algorithms for high dynamic range video. In *Computer Graphics Forum*, volume 36, pages 565–592. Wiley Online Library, 2017. 3

[12] Michael T. Emmerich and André H. Deutz. A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. 17(3):585–609, Sept. 2018. 4

[13] Rafael C. Gonzales and Richard E Woods. *Digital image processing*. Prentice hall New Jersey, 2002. 2

[14] Mohit Gupta, Amit Agrawal, Ashok Veeraraghavan, and Srinivasa G Narasimhan. Structured light 3d scanning in the presence of global illumination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 713–720, 2011. 17

[15] Nikolaus Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 2389–2395, July 2009. 3

[16] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. https://doi.org/10.5281/zenodo.2559634, Feb. 2019. 3, 12, 13

[17] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996. 3

[18] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. 3

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (CVPR)*, pages 2961–2969, 2017. 18

[20] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research*, 15(1):3735–3739, 2014. 12, 13

[21] Jun Nishimura, Timo Gerasimow, Rao Sushma, Aleksandar Sutic, Chyuan-Tyng Wu, and Gilad Michael. Automatic ISP image quality tuning using nonlinear optimization. In *IEEE International Conference on Image Processing (ICIP)*, pages 2471–2475, 2018. 17

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. 16

[23] 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 (SIGGRAPH)*, 38(4):27, 2019. 16, 17, 19

[24] 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. 2

[25] Lucie Yahiaoui, Ciarán Hughes, Jonathan Horgan, Brian Deegan, Patrick Denny, and Senthil Yogamani. Optimization of ISP parameters for object detection algorithms. *Electronic Imaging*, 2019(15):44–1, 2019. 20