Hyperparameter Optimization in Black-box Image Processing using Differentiable Proxies

ETHAN TSENG, FELIX YU, and YUTING YANG, Princeton University, United States
FAHIM MANNAN and KARL ST. ARNAUD, Algolux, Canada
DEREK NOWROUZEZAHRAI, McGill University, Canada
JEAN-FRANÇOIS LALONDE, Université Laval, Canada
FELIX HEIDE, Princeton University, United States and Algolux, Canada

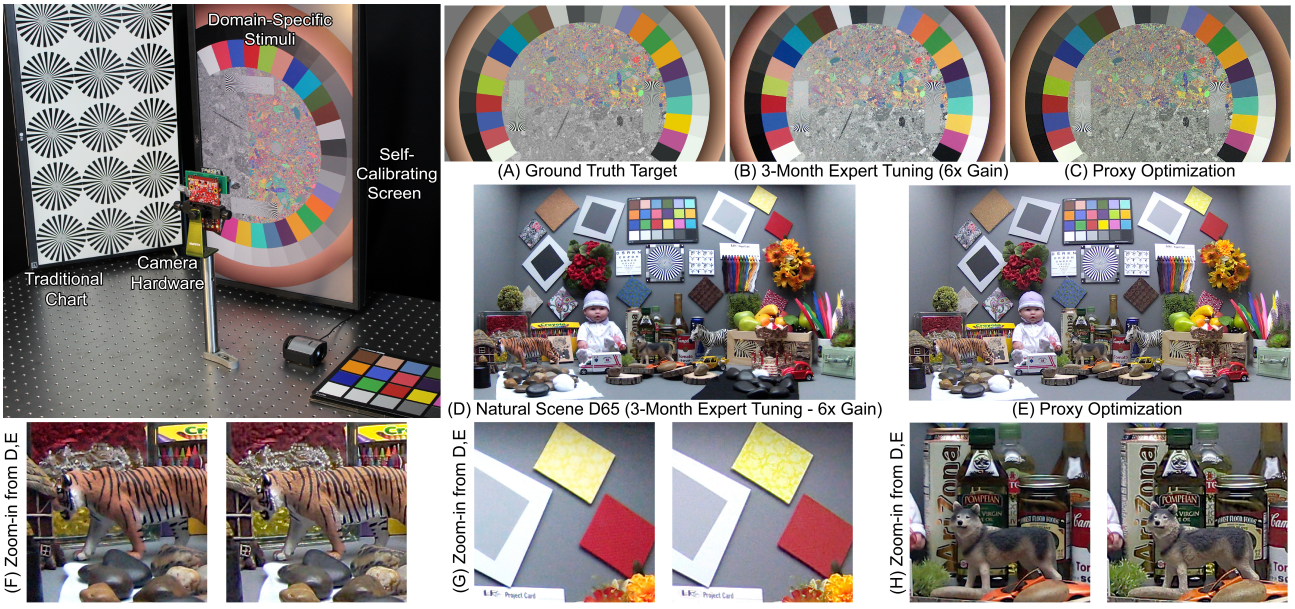


Fig. 1. We combine a radiometrically calibrated screen with a *hardware-in-the-loop* camera to generate stimulus/processed image pairs. We then train a *differentiable proxy model* on the displayed target (A), before using the model to optimize application-specific imaging tasks. Here, we optimize hyperparameters of a hardware ISP (ARM Mali C71) for perceptual accuracy. Our prototype improves image quality (C & E), compared to a three-month-long tuning by imaging experts (B & D). Our method is automatic and generates results in under an hour, enabling rapid prototyping for domain-specific imaging systems.

Nearly every commodity imaging system we directly interact with, or indirectly rely on, leverages power efficient, application-adjustable black-box hardware image signal processing (ISPs) units, running either in dedicated hardware blocks, or as proprietary software modules on programmable hardware. The configuration parameters of these black-box ISPs often have complex interactions with the output image, and must be adjusted prior to deployment according to application-specific quality and performance metrics. Today, this search is commonly performed *manually* by "golden eye" experts or algorithm developers leveraging domain expertise. We present a *fully automatic* system to optimize the parameters of black-box hardware and software image processing pipelines according to any arbitrary (i.e., application-specific) metric. We leverage a *differentiable* mapping between the configuration space and evaluation metrics, parameterized by a convolutional neural network that we train in an end-to-end fashion with imaging hardware in-the-loop. Unlike prior art, our *differentiable proxies* allow for high-dimension parameter search with stochastic first-order optimizers, without explicitly modeling any lower-level image processing transformations. As such, we can efficiently optimize black-box image processing pipelines for a variety of imaging applications, reducing application-specific configuration times from months to hours. Our optimization method is fully automatic, even with black-box hardware in the loop. We validate our method on experimental data for real-time display applications, object detection, and extreme low-light imaging. The proposed approach outperforms manual search qualitatively and quantitatively for all domain-specific applications tested. When applied to traditional denoisers, we demonstrate that—just by changing hyperparameters—traditional algorithms can outperform recent deep learning methods by a substantial margin on recent benchmarks.

CCS Concepts: • **Computing methodologies → Computational photography**.

Additional Key Words and Phrases: image processing

ACM Reference Format:
Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl St. Arnaud, Derek Nowrouzezahrai, Jean-François Lalonde, and Felix Heide. 2019. Hyperparameter Optimization in Black-box Image Processing using Differentiable

0730-0301/2019/7-ART27 $15.00
https://doi.org/10.1145/3306346.3322996

Proxies. *ACM Trans. Graph.* 38, 4, Article 27 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3322996

1 INTRODUCTION

We routinely interact with commodity imaging systems. The growing ubiquity of customizable image signal processors (ISPs) underlying these systems is evidenced in their varied uses: communication, entertainment, personal security, surveillance, and emerging applications such as autonomous driving. Power and compute efficient hardware ISPs transform signals captured by imaging sensors into images suitable for either human or automated consumption (e.g., in computer vision). Hardware ISPs are orders of magnitude faster, cheaper, and more power-efficient than software image processing pipelines, albeit less programmable. Many applications, e.g. safety-critical real-time robotic systems, require real-time imaging systems to process almost double-digit megapixel streams without frame drops, e.g. Sony IMX324, mandating such hardware ISPs.

Most hardware ISPs are black-box units: their behavior is configurable according to a set of user-adjustable hyperparameters, however the details of their inner-workings are not usually revealed to the user. Emerging software-based ISPs allow for some programmability [Barry et al. 2015], while individual accelerator blocks or custom software algorithms are still often proprietary black-box systems. As such, a common design workflow involves the adjustment of the hyperparameters in order to (ideally) maximize an application-specific performance metric, such as texture accutance [Baxter et al. 2012], modulation transfer function (MTF), or contrast detection probability for computer vision applications [Stead 2016]. Unfortunately, many factors complicate this ISP parameter adjustment problem: the subcomponents that form the ISP pipeline are typically non-differentiable; and, for researchers, the exact forms of these subcomponents are typically well-guarded trade secrets, purposefully obfuscated from the users performing the parameter adjustment.

The standard industry practice relies on so-called "golden eye" experts to adjust the parameters *manually* (i.e., by hand) while combining traditional key performance index (KPI) metrics with visual inspection across a necessarily small dataset of standardized natural images. The downsides of a such a manual process are that it is both extremely time consuming (i.e., on the order of several man-months of work per problem instance), and that there is no guarantee that the hyperparameters resulting from the incremental iterative process are optimal in any local or global measure. Moreover, this process is only feasible for visual tasks and for white-box systems. With only black-box implementations available, experts cannot rely on domain knowledge given the obscured nature of the parameters. For higher-level vision analytic tasks, such as object detection, it is unclear how to tune the ISP in an optimal way, and hence experts do not exist. Work on automating this process has focused on optimizing individual white-box algorithmic blocks of software ISPs with 0^{th}-order parameter search methods. Unfortunately, these solutions do not scale to the high-dimensional parameter spaces of 30–50 parameters or more of modern ISPs, do not apply to closed-source or hardware systems available, and cannot jointly optimize all parameters of the mixed categorical and continuous variable set.

In this paper, we propose a fully automatic method for optimizing black-box ISPs. To do so, we model the ISP with a *differentiable proxy function* that *learns* to reproduce the entire ISP image transformation process as a function of its input configuration parameters. In practice, we parameterize our *differentiable proxy functions* using convolutional neural networks (CNN) and, for any given hardware ISP, we automatically generate training data using a hardware-in-the-loop system. This system creates training pairs by displaying control images on a calibrated high-resolution monitor, setting ISP configuration parameters, and recording the ISP-processed output image. We perform a one-time calibration to ensure accurate pixel alignment between the input and output images, before training our differentiable proxy functions in a supervised learning fashion.

We apply differentiable proxy functions to several domain-specific problems. These include optimizing human-viewable outputs for specific imaging systems according to tailored perceptual metrics, and optimizing imaging system output for consumption by secondary computer vision systems for, e.g., pedestrian and car detection in autonomous driving scenarios, and low-light imaging by tuning traditional denoising algorithms.

Our approach presents several advantages over the state-of-the-art: first, it is fully automatic, requiring no human intervention; second, calibrated screen-based stimuli generalize to unseen testing scenarios without the need for custom datasets (i.e., those captured *in the wild* or annotated by hand); third, differentiable proxy functions scale to optimization problems with discrete and continuous trainable hyperparameters without knowing their specific function in the processing pipeline; lastly, we model the entire imaging system in an *end-to-end* manner. We make the following contributions:

- the first automatic, scalable hyper-parameter optimization method for black-box imaging systems;
- we introduce *differentiable proxy functions* to model arbitrary *black box* imaging systems;
- a *hardware-in-the-loop* setup to train the differentiable proxy functions for parameter optimization *and* benchmarking tasks according to application-specific performance metrics;
- we analyze and benchmark against experimental data for several applications, including automotive object detection and extreme low-light smartphone imaging. Our approach outperforms manual search qualitatively and quantitatively on *all* domain-specific applications tested. When applied to existing denoisers, we demonstrate that—just by changing hyperparameters—these algorithms can outperform recent deep image processing methods by a substantial margin on recent low-light benchmarks.

Our prototype is compatible with *any* imaging system due to the hardware-in-the-loop nature of our calibration and training, without any knowledge of the imaging system (i.e., regardless of whether the form or function of the ISPs constituent parts are known, let alone differentiable). This is essential for end-to-end training, utility and scalability for many real-world applications.

2 RELATED WORK

We review prior art most related to our contributions, below.

Camera Image Processing Pipelines. Raw measurements from digital camera sensors are degraded by many factors, including photon shot noise, read-out noise, optical aberrations, sub-sampling on color-filter arrays, and cross-talk. Recovering a latent high-quality

image from these measurements on modern camera modules mandates complex low-level image processing pipelines, typically implements with highly-optimized ASICs [MT9P111 2015; Ramanath et al. 2005; Shao et al. 2014; Zhang et al. 2011]. Moreover, the demands for real-time processing of high throughput and high resolution image *sequences* further motivate the use of such custom hardware solutions, e.g., commodity smartphones can capture 4K video at ≥ 30 FPS and the next-generation of cameras will soon support slow-motion video capture in HD. In sharp contrast to portrait capture applications, allowing for seconds per photo, these real-time applications require immediate processing — multi-capture processing, e.g. burst imaging [Hasinoff et al. 2016] in low light settings, are not possible if per frame capture and compute exceed hundreds of milliseconds.

These performance and throughput constraints apply outside of photographic applications. Imaging systems for driver assistant systems, fully-autonomous vehicles or other robotic usage scenarios, for example, require real-time reaction and often with only enough time to capture two or three sequential HDR images. Such constraints have lead to the emergence of split-pixel sensors (e.g., OmniVision OV10640, OV10650) and domain specific ISPs (e.g., ARM Mali C71). These stringent performance constraints lead to challenging algorithmic and hardware design constraints, and so most hardware ISPs are proprietary designs with highly optimized processing modules. Here, control of the underlying processing units are only exposed through a (potentially large) set of hyperparameters, chosen often by the hardware vendors and not, e.g., application developers.

Alternative ISP designs are today restricted to off-line processing tasks. For example, Heide et al. [2014] pose low-level image processing as an optimization problem, which is computationally intensive and an order of magnitude slower than real-time ISPs. Recent data-driven approaches address individual subproblems of the image processing pipeline with deep neural networks, such as demosaicking [Gharbi et al. 2016], tonemapping [Gharbi et al. 2017], low-light denoising [Chen et al. 2018] and other operators [Chen et al. 2017; Fan et al. 2018; Xu et al. 2015]. These methods substantially improve runtime performance but are still orders of magnitudes slower than hardware ISPs, requiring on the order of a second to process a 12 Megapixel image on a >100 Watt desktop GPU [Gharbi et al. 2016], precluding high-resolution or real-time processing. One interesting property of data-driven methods, however, is that they can be automatically optimized for specific tasks, while standard ISPs require application-specific manual tuning. More generally, works in graphics substitute complex simulation models with more efficient or flexible proxies. NeuroAnimator [Grzeszczuk et al. 1998] models complex physics-based simulations with neural networks, whereas deep networks have also been used to supplement [Chaitanya et al. 2017], replace [Li et al. 2018] or invert [Liu et al. 2019] image synthesis pipelines.

Our work bridges the performance of hardware and data-driven ISPs by addressing this problem, *efficiently* and *automatically* optimizing black-box hardware ISPs capable of delivering application-tailored real-time performance with high power efficiency.

Image Quality Analysis. The complexity of the Human Visual System (HVS), and that of modern imaging systems, both contribute to artifacts in end-to-end capture-to-consumption imaging pipelines. Image Quality (IQ) assessment of camera systems and ISPs is a complex and evolving field [ISO [n. d.]a,n; Phillips and Eliasson 2018] that aims to quantify application-specific quality measures. Another complication in the design of reliable and useful IQ metrics is the fact that (even geographically-dependent) users may prefer images that are "artificially" enhanced, e.g., with boosted chroma.

Typical spatial IQ metrics focus on one aspect of image quality, such as sharpness, noise or blur [Baxter et al. 2012]. Measuring a key performance indicator (KPI) typically involves performing captures of printed paper or transmissive calibration charts under controlled lighting, computing statistics from one or more ROI (Region Of Interest) in the measured output images. In order for these statistics to yield meaningful quality metrics, not only are the charts and capture conditions standardized, but normalization procedures (i.e., subtracting noise estimates obtained on a flat patch from the power spectrum of a textured area, or inverting the tone map) are mandated. Even with a well-calibrated KPI, better performance on the metrics may not necessarily correlate with improved perceptual quality: e.g., excessively denoised images tend to appear "plastic-y" and unrealistic. As a result, any claim of "good image quality" can at best be attributed to scoring sufficiently well on many KPIs, each of which is ideally correlated with a different (meaningful, application-tailored) artifact. Even then, subjective evaluation by a "golden eye" expert and field testing are mandated for quality assurance.

Automated Camera Design. While no end-to-end automated design tools exist for ISP pipelines, the design of other components of the imaging stack have been automated, including optics, electronic integrated circuits ISP controllers and camera controllers. Optical systems are designed to minimize optical aberrations, i.e., deviations from a perfect linear optics model [Gauss 1843], under the constraints of available footprint, cost, and lens element types/shapes (e.g., spherical elements). State-of-the-art optical design software, such as Zemax [Geary 2002] or Code5 V [Garrard et al. 2005] can optimize the surface profiles and types of refractive lenses. These tools use a mid-level metric, a so-called merit function, which typically strikes a compromise between a variety of criteria [Malacara-Hernández and Malacara-Hernández 2016]: trading off the point-spread-function (PSF) shape across sensor locations, lens configurations (e.g., zoom levels), and target wavelength bands. Recent work has explored automated design of diffractive optical elements [Sitzmann et al. 2018; Stork and Gill 2013, 2014] with an image quality metric loss instead of an intermediate merit function. Note that, here, simplified design tasks are employed due to the complex, non-differentiable behavior of modern ISPs. We address this limitation, training *differentiable proxies* of hardware ISPs with a calibrated, hardware-in-the-loop methodology.

Recently, Nishimura et al. [2018] optimize individual blocks of white-box software ISPs using 0^{th}-order Nelder-Mead. Their approach differs in a number of important ways from the proposed method. While the proposed system supports hardware ISPs with more than 30 jointly trained discrete and continuous parameters, the 0^{th}-order search in [Nishimura et al. 2018] is limited to 3-4 continuous parameters per block. As a result, this approach can only be applied per ISP block (which has less than a hand-full of continuous parameters), requires the knowledge of the blocks and

their role, and does not allow for joint optimization. The authors are therefore limited to a software ISP with known nodes (that have a few continuous parameters), which are sequentially optimized. Note that hardware ISPs, including those in this work, may actually have dozens of discrete and continuous parameters. Higher-level applications that require first-order training on large datasets, as well as hardware ISPs cannot use [Nishimura et al. 2018].

0^{th}-order Optimization. Efficient and effective non-differentiable black-box optimization remains an open research problem with a variety of solutions. For low-dimensional problems and where fast function evaluation is an option, grid search is a commonly used strategy in practice [Bergstra and Bengio 2012]. The cost of this method grows exponentially with input dimension, making it impractical for larger problems.

Early work on 0^{th}-order optimization methods for real-valued relaxations rely on geometric search procedures. Powell's widely adopted method [Powell 1965] applies bisection search, and the Nelder-Mead method [Nelder and Mead 1965] relies on various geometric simplexes to define its search region. More recently, random search methods have been proposed for hyperparameter optimization problems with mixed categorical and continuous parameters [Bergstra and Bengio 2012]. Instead of pure stochastic search, these methods build a probabilistic prior over a history of function evaluations, using this probabilistic proxy to approximate the objective landscape and update subsequent random samples. Bayesian optimization methods differ in their modeling of the surrogate functions they rely on [Bergstra et al. 2013; Shahriari et al. 2016; Snoek et al. 2012; Swersky et al. 2013], but operate similarly on a high level. While domain-agnostic Bayesian optimization methods are challenging to scale to higher dimensions, we show that domain-specific proxies allow us to efficiently solve high-dimensional hyperparameter optimization problems for image processing pipelines. Evolutionary algorithms, such as [Hansen et al. 2003; Loshchilov et al. 2017], can scale to higher-dimensional spaces but require objectives that are efficient to evaluate for the generation of large populations. This prohibits loss functions defined over large training datasets.

3 THE IMAGING PIPELINE AND ITS PARAMETERS

ISPs are composed of many processing stages, most generally converting an incoming light field into a (potentially displayable) image. We briefly review the most common ISP stages, and their associated parameters, keeping in mind that specialized parameter settings are necessary to obtain a desirable application-specific output.

3.1 Common Imaging Pipeline Stages

We decompose typical imaging pipelines into the following stages:

(1) **Optics**: first, a lens (or more general optical system) focuses the light field onto an array of photo-diodes, which subsequently converts the photon flux of incoming irradiance to RAW digital values via analog-to-digital converter circuits.
(2) **White Balance & Gain**: after removing the black level bias, correcting for defect pixels and vignetting, the RAW pixel values are color-corrected and gain-adjusted according to the (preset or estimated) illuminant color [Ramanath et al. 2005].
(3) **Demosaicking**: since the RAW pixel values are most commonly captured with a Bayer mosaic (alternating R-G-G-B 2×2 color filter array), trichromatic RGB values are reconstructed, for example by interpolation [Zhang et al. 2011].
(4) **Denoising**: resulting RGB values are filtered to attenuate sensor noise, for example with edge-preserving filters (Bilateral Filter [Choi et al. 2014; Tomasi and Manduchi 1998]) or even non-local patch matching [Dabov et al. 2007; Zhang et al. 2016].
(5) **Color & Tone Correction**: several adjustments can be performed afterwards to improve overall image appearance. For example, global operations (applying a gamma curve, adjusting image contrast via histogram manipulation), or local operations (emphasizing edges with sharpening) can be applied here.
(6) **Colorspace Conversion & Compression**: pixel values are converted to a specific colorspace (e.g. to sRGB) before compression (e.g. to jpg), storage, or further processing.

We model stages two to six, assuming RAW sensor pixel measurements as input. While one could incorporate potentially parametrized optics (stage one) in our framework, we choose to simplify our prototype and fix the optics without tunable parameters.

3.2 Imaging Pipeline Hyperparameters

Since any specific ISP hyperparameterization depends on the parameterization of, i.e. algorithms employed at each stage, the exact number of hyperparameters will vary across ISPs. Table 1 provides one such example, the ARM Mali-C71 ISP we use in our experiments. This ISP is a next-generation (pre-mass-production) state-of-the-art hardware ISP for real-time applications in robotics, consumer devices and automotives. It has a **32-dimensional** hyperparameter space that can be tuned to the application-specific needs of a user.

Optimizing such an ISP according to an application-specific performance metric is challenging. In addition to the complexity introduced by the large number of hyperparameters, the relationship between any single parameter's variation and the output performance metric is typically a complex, non-linear function. Fig. 2 visualizes the behavior of example performance metrics as a function of the value of two hyperparameters from the ARM Mali-C71 ISP. As such, the *coupled* sensitivity of the performance to changes across *many* hyperparameters is complex and highly non-linear. Unfortunately, traditional 0^{th}-order search methods are ineffective on high-dimensional, non-convex, non-differentiable error landscapes. Similarly, grid-based search cannot scale to the high-dimensional spaces of such real-world ISPs. As such, experts rely on traditional heuristics and domain knowledge to manually adjust parameters.

4 DIFFERENTIABLE PROXY MODELS

We now describe an end-to-end model for arbitrary ISP pipelines (ignoring optics, Sec. 4.1) using a differentiable proxy function (Sec. 4.2) parameterized by a single convolutional neural network (CNN; Sec. 4.3). This will allow us to leverage existing 1^{st}-order methods to efficiently perform application-specific ISP hyperparameter optimization. We then specialize our methodology to both *software-based* (Sec. 5) and *hardware-in-the-loop* realizations (Sec. 6).

We consider only those tunable ISP stages succeeding the optical system and sensor readout. We will assume, moving forward, that an

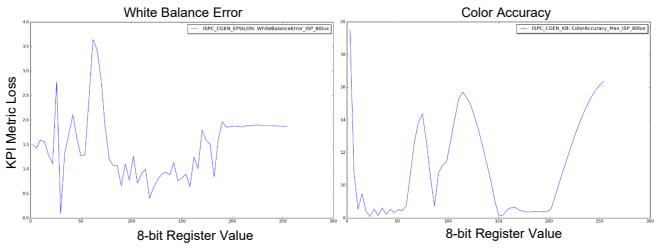


Fig. 2. Traditional image analysis metrics as a function of varying one ISP hyperparameter value at a time. We show values for a white balance accuracy and color accuracy metric [Koren 2006]. Changing the values of two hyperparameters, controlling demosaicking and tonemapping, results in metric values with highly rugged, non-differentiable objective landscapes.

ISP expects an image (sRGB or RAW) as input, and outputs processed image data for either human or automated consumption.

4.1 ISP Modeling

We model ISPs as functions f_{ISP} that map an input image $\mathbf{I}$ to an output/processed image $\mathbf{O}_{\text{ISP}}$. ISPs are parameterized by hyperparameters $\mathcal{P}$, and so we denote $\mathbf{O}_{\text{ISP}} = f_{\text{ISP}}(\mathbf{I};\mathcal{P})$. We are interested in solving *hyperparameter optimization problems* of the form

$$\mathcal{P}^* = \arg\min_{\{\mathcal{P}\}} \sum_{i=1}^{N} \mathcal{L}_{\text{TASK}}\left(f_{\text{ISP}}(\mathbf{I}_i;\mathcal{P}),\mathbf{T}_i\right) , \tag{1}$$

where we seek a set of *optimal ISP parameter settings* $\mathcal{P}^*$ by optimizing a task-specific performance metric. We model this as a loss function $\mathcal{L}_{\text{TASK}}$ that measures performance deviation between the ISP output and N known "control" target images $\mathbf{T}_i$.

4.2 Optimizing Black-box Imaging Systems Hyperparameters with Differentiable Proxies

We propose the following 2-stage procedure to minimize Eq. 1, that is, to recover the hyperparameters of a black-box ISP for a particular task. Our approach relies on a Differentiable Proxy Function (DPF) that learns to reproduce the behavior of a given ISP.

(1) **Training the DPF to learn the ISP function**: the DPF is trained to approximate the function applied by the ISP on an input image, as a function of its hyperparameters;
(2) **Optimizing the ISP hyperparameters with the trained DPF**: the learned DPF is then used to recover the best ISP hyperparameters for a given task.

Each of these stages is described in more details below.

4.2.1 Stage 1: Training the DPF to learn the ISP function. In a general setting, f_{ISP} is either unknown (as with a black-box system) or non-differentiable which, combined with the high dimensionality of $\mathcal{P}$, increases the challenge of solving Eq. 1. Here, we assume the only operation available to us is the ability to "evaluate" f_{ISP} on example input images and parameter settings.

In order to solve Eq. 1 under such constraints, we introduce a representation f_{PROXY} that will allow us to solve it *by proxy*. We will *train* f_{PROXY} so as to mimic the *functional behavior* of f_{ISP} as closely as possible (see Fig. 3). As with f_{ISP}, this proxy model also maps input images to output images, however it also expects ISP hyperparameters $\mathcal{P}$ as *additional* inputs.

We parameterize f_{PROXY} by its own weights $\mathcal{W}$ and constrain the form of our proxy to those that are *differentiable* w.r.t. $\mathcal{P}$ and $\mathcal{W}$,

$$\mathbf{O}_{\text{proxy}} = f_{\text{PROXY}}(\mathbf{I},\mathcal{P};\mathcal{W}) , \tag{2}$$

where we can evaluate f_{PROXY} and $\partial f_{\text{PROXY}}/\partial p_n$, for every $p_n \in \mathcal{P}$ (equivalently for every $w_n \in \mathcal{W}$).

Indeed, if f_{PROXY} is differentiable and sufficiently expressive to approximate an arbitrary, black-box f_{ISP}, then we can optimize for weights $\mathcal{W}$ that yield a proxy representation that behaves similarly to the ISP. Specifically, we seek a proxy representation that is able to produce $\mathbf{O}_{\text{proxy}} \approx \mathbf{O}_{\text{ISP}}$ for any combination of input image $\mathbf{I}$ and hyperparameters $\mathcal{P}$. To obtain this representation, we solve the following optimization problem for the proxy weights

$$\mathcal{W}^* = \arg\min_{\{\mathcal{W}\}} \sum_{i=1}^{M} \| f_{\text{PROXY}}(\mathbf{I}_i,\mathcal{P}_i;\mathcal{W}) - f_{\text{ISP}}(\mathbf{I}_i;\mathcal{P}_i)\|_2 , \tag{3}$$

over the M images and sampled hyperparameters. Here, the L2 loss ensures that the proxy function image $\mathbf{O}_{\text{PROXY}}$ is as close as possible to that produced by the ISP $\mathbf{O}_{\text{ISP}}$. Since f_{PROXY} is differentiable with respect to $\mathcal{W}$, we can employ 1st-order techniques to solve Eq. 3.

4.2.2 Stage 2: Optimizing ISP Hyperparameters with Differentiable Proxies. Solving Eq. 3 yields weights $\mathcal{W}^*$ that we use to instantiate a relaxed instance of the hyperparameter search problem (Eq. 1),

$$\mathcal{P}^* = \arg\min_{\{\mathcal{P}\}} \sum_{i=1}^{N} \mathcal{L}_{\text{TASK}}\left(f_{\text{PROXY}}(\mathbf{I}_i,\mathcal{P};\mathcal{W}^*),\mathbf{T}_i\right) . \tag{4}$$

Here, substituting f_{ISP} with f_{PROXY} from Eq. 1 to Eq. 4 results in a more tractable problem, since we can not only evaluate f_{PROXY} but also any of its partial derivatives with respect to the (proxy) ISP hyperparameters, $\partial f_{\text{PROXY}}/\partial p_n$. This allows us to employ modern 1st-order optimization techniques, including variants of stochastic

Table 1. Parameters for each stage of the ARM Mali-C71 ISP we use in our experiments. This ISP has a total of 32 hyperparameters to tune. It took several "golden eye" experts several months to manually tune the parameter settings (included in the table for completeness) for a visual tuning application, while the proposed system requires less than one hour.

White Balance

Parameter	Value	Max
gain 00	583	2^{12}
gain 01	271	2^{12}
gain 11	587	2^{12}

Demoisaicking

Parameter	Value	Max
vh slope	190	2^8
vh thresh	220	2^{12}
va slope	175	2^8
va thresh	210	2^{12}
aa slope	170	2^8
aa thresh	100	2^{12}
uu slope	165	2^8
uu thresh	210	2^{12}
sharp alt ld	45	2^8
sharp alt ldu	45	2^8
sharp alt lu	25	2^8
fc alias slope	85	2^8
fc alias thresh	0	2^8
fc slope	130	2^8
np offset	3	2^8

Denoising

Parameter	Value	Max
thresh 1h	5	2^8
strength 1	190	2^8
thresh 4h	10	2^8
strength 4	255	2^8
thresh long	48	2^8

Color & Tone Correction

Parameter	Value	Max
lut knee	207	2^8
lut power	129	2^8
lut shadow	31	2^8

Colorspace Conversion

Parameter	Value	Max
coef a 11	4461	5880
coef a 12	4001	5880
coef a 21	4068	5880
coef a 22	4398	5880
coef a 31	4135	5880
coef a 32	3842	5880

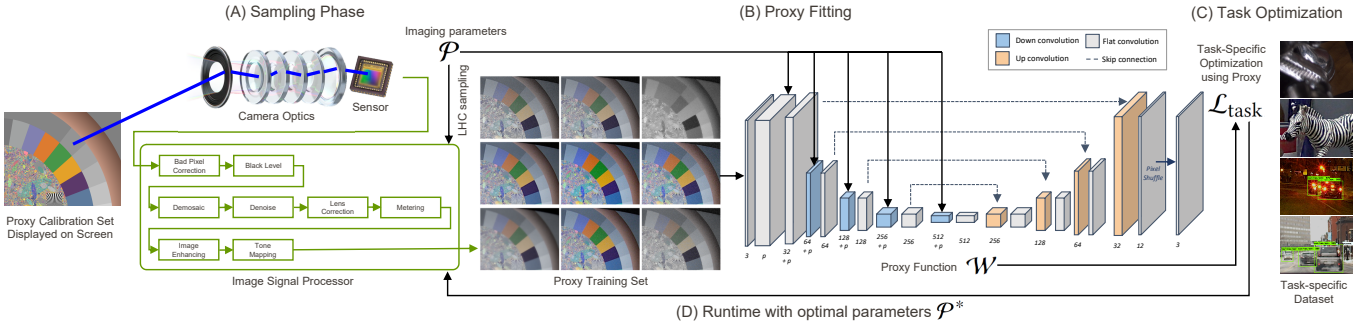


Fig. 3. Proposed hyper-parameter optimization stages and proxy architecture. a) latin-hypercube (LHC) random sampling b) Training on LHC random samples, c) Fixing learned parameters and optimizing input parameters or image itself given a separate loss function either on traditional KPIs or with separate input/output dataset. d) Runtime execution using the hardware architecture with the optimized parameters.

gradient descent, where backpropagation allows us to now compute partial derivatives of arbitrary (i.e., task-specific) performance losses with respect to the (proxy) ISP hyperparameters, $\partial \mathcal{L}_{\text{TASK}}/\partial p_n$, for every $p_n \in \mathcal{P}$. These techniques scale more robustly to larger hyperparameter spaces and non-linearities.

4.3 Differentiable Proxies: Form & Parameterization

Deep convolutional neural networks (CNNs) are a natural architectural choice for differentiable proxy functions f_{PROXY}: they have proven both flexible and powerful in a variety of image-based tasks [Ulyanov et al. 2017], well-established techniques exist to train such models in an end-to-end fashion, and we can naturally impose differentiability for the hyperparameteric inputs to the model.

In practice, we employ a variant of the UNet CNN architecture ([Ronneberger et al. 2015]; see Fig. 3) with {32, 64, 128, 256, 512} encoder channels (and vice versa for the decoder), with minor variations depending on whether we employ a software- or hardware-based training regime (detailed in Secs. 5 and 6, respectively).

Mirroring the localized processing behavior of modern ISPs, as well as echoing well-established practice in the machine learning and computer vision literature (i.e., [Isola et al. 2017]), our CNN operates on local image patches (specifically of 512×512 resolution). In addition to the patch input channels (e.g. three for sRGB input), we concatenate as many channels as there are hyperparameters $\mathcal{P}$ in the ISP, where each channel is simply the value of the hyperparameter replicated over the spatial dimension. For example, a proxy CNN modeling a 32-parameter ISP taking as input an sRGB image, would receive a 35-channel (32+3) input patch. To encourage the model to learn the effects of the hyperparameters on the output image, we further append the input hyperparameter channels to each downsample layer (shown as blue layers in Fig. 3). To avoid introducing unwanted dependencies between channel layers, each up-/down-sampling operation is performed independently per channel. The hyperparameter inputs are normalized to lie in the $[0, 1]$ interval for more stable training. We train the model using latin-hypercube (LHC) random sampling of the ISP hyperparameters. Application-specific details are provided in Sec. 5 and 6.

5 SOFTWARE VALIDATION EXPERIMENTS

We first validate the utility and effectiveness of differentiable proxy functions on challenging imaging problems using existing black-box software processing blocks, before presenting our *hardware-in-the-loop* end-to-end ISP hyperparameter optimization setup (Sec. 6). All of these experiments use the UNet architecture from Sec. 4.3, trivially adapted to expect a 3-channel sRGB 512×512 image subpatch as input, and produce a 512×512 output.

5.1 Low-light Black-box Denoising Task

We train a proxy f_{PROXY} to mimic the behavior of the BM3D software denoiser [Dabov et al. 2007] (i.e., f_{ISP}) on a standardized low-light denoising task, using the SIDD sRGB dataset [Abdelhamed et al. 2018] which contains noisy/clean natural images pairs acquired from five different smartphone cameras. We use the original black-box obstructed MATLAB binaries provided by the authors [Dabov et al. 2007], and we set the standard deviation value using standard shifted difference estimation (see supplemental document). We optimize the Wiener transform category, neighborhood size, patch size, a linear scale for sigma, and aggregation color space. These parameters are both categorical and continuous (see the supplemental document).

We extract $M = 6900$ random 512×512 image patches across the entire noisy image dataset to serve as input training images $\mathbf{I}_{\text{NOISY}}$. We then execute the BM3D denoiser on each patch with randomly selected hyperparameters values to obtain the corresponding $\mathbf{O}_{\text{ISP}}$. We optimize for the proxy weights $\mathcal{W}$ with the ADAM optimizer on this training set (see the supplemental document).

Once trained, we apply our trained proxy to this sRGB denoising instance of the hyperparameter optimization problem, in order to obtain optimal BM3D parameters. Specifically, we now rely on both the noisy image patches $\mathbf{I}_{\text{NOISY}}$ as well as their ground truth denoised counterparts $\mathbf{I}_{\text{CLEAN}}$, from the SIDD sRGB training dataset. The BM3D parameters are obtained as the result of solving the following task-specific specialization of Eq. 4,

$$\mathcal{P}^*_{\text{BM3D}} = \underset{\{\mathcal{P}_{\text{BM3D}}\}}{\arg\min} \sum_{i=1}^{N} \| f_{\text{PROXY}}(I_{\text{NOISY}}, \mathcal{P}_{\text{BM3D}}; \mathcal{W}^*) - \mathbf{I}_{\text{CLEAN}} \|_2 \ , \quad (5)$$

where we minimize the mean squared error between the clean, ground truth images $\mathbf{I}_{\text{CLEAN}}$ and the output of our proxy.

Table 2 summarizes the quantitative results of this black-box denoising experiment. Our proxy-optimized BM3D ISP substantially outperforms all other alternatives, yielding new state-of-the-art results for sRGB-to-sRGB denoising on this dataset. Fig. 4 illustrates some representative qualitative results on this task. These results

Table 2. Quantitative sRGB-to-sRGB denoising results on the SIDD sRGB dataset [Abdelhamed et al. 2018]. Previous benchmark leader indicated in light blue. The dataset uses ground truth, stored on an external server and hidden from the user, to ensure fairness. Our method finds better hyperparameters for the existing BM3D algorithm, significantly outperforming the state-of-the-art. Corresponding qualitative results are shown in Fig. 4.

Method		PSNR	SSIM
Proxy-opt. BM3D	(Ours)	**34.34**	**0.911**
CBDNet	[Guo et al. 2018]	33.28	0.868
KSVD-DCT	[Elad and Aharon 2006]	27.51	0.780
KSVD-G	[Elad and Aharon 2006]	27.19	0.771
EPLL	[Zoran and Weiss 2011]	27.11	0.870
KSVD	[Aharon et al. 2006]	26.88	0.842
NLM	[Buades et al. 2005]	26.75	0.699
WNNM	[Gu et al. 2014]	25.78	0.809
BM3D	[Dabov et al. 2007]	25.65	0.685
FoE	[Roth and Black 2005]	25.58	0.792
TNRD	[Chen and Pock 2017]	24.73	0.643
MLP	[Burger et al. 2012]	24.71	0.641
GLIDE	[Talebi and Milanfar 2014]	24.71	0.774
LPG-PCA	[Zhang et al. 2010]	24.49	0.681
DnCNN	[Zhang et al. 2017]	23.66	0.583
DemosaicNet[1]	[Gharbi et al. 2016]	22.38	0.369

highlight the importance of the hyperparameter optimization problem and the impact that better hyperparameter settings can have on task-specific performance, even when relying on a fixed ISP.

Optimization Performance & Alternatives. Our proxy-based hyperparameter optimization completes in under three (3) hours (on a single, modern desktop GPU), including the time to train the differentiable proxy. This is owed, in part, to the effectiveness of 1^{st}-order optimization methods enabled by the differentiable nature of our proxies. It is not feasible to rely, alternatively, on 0^{th}-order techniques commonly applied to this problem, such as Bayesian optimization methods: these methods would require evaluating the BM3D ISP once per 6900 training image patches per iteration, leading to roughly 5×10^3 seconds or one hour *per iteration*. These methods typically require roughly 500 iterations for image processing tasks, and so a total optimization time of about **21 days**.

5.2 Validation

Next, we demonstrate the flexibility of DPFs on distinct hyperparameter optimization problems: setting the hyperparameters of a global tone mapping operator (Sec. 5.2.1) and general, noise-polluted non-linear optimization benchmark functions (Sec. 5.2.2).

The validation test results demonstrate the applicability of proxy models to efficient global image operator optimization (i.e., beyond local demosaicking or per-pixel operations), and their ability to scale to challenging high-dimensional non-linear search problems.

5.2.1 Global Image Tonemapping. To validate the applicability of DPFs to different scenarios such as global image processing operations, we train a proxy to reproduce the behavior of a global tonemapping operator, f_{TONEMAP} [Paris et al. 2011].

[1]We use an unmodified DemosaicNet (trained on its original RAW dataset) and evaluate it on RAW using its default parameters. Note that all other methods map sRGB to sRGB.

Here, we set the proxy input space to match that of the tonemapping operator, $f_{\text{PROXY}}(\mathbf{I}_i, \alpha, \beta, \epsilon; \mathcal{W})$, where α and β control the form of an underlying Laplacian filtering pyramid and ϵ is a numerical stability parameter [Paris et al. 2011].

Outside of differences in the parameterization of the problem, we proceed similarly to Sec. 5.1: we generate $M = 10^4$ random 512×512 input/output image patches extracted from a dataset of 1,000 natural images ($\mathbf{I}_{\text{IN}}$ and $\mathbf{O}_{\text{TONEMAP}} = f_{\text{TONEMAP}}(\mathbf{I}_{\text{IN}}; \alpha, \beta, \epsilon)$) with randomly selected hyperparameters $\{\alpha, \beta, \epsilon\}$, and we optimize the proxy weights $\mathcal{W}$ with the Adam optimizer (training details are in the supplementary document), solving Eq. 3 with $\mathcal{P} \equiv \{\alpha, \beta, \epsilon\}$.

We then manually select user-defined values for $(\alpha, \beta, \epsilon)$, and tonemap all the training images with these fixed hyperparameters. Finally, we use the trained proxy to estimate the hyperparameters from the set of tonemapped images. As in Sec. 5.1, we employ an L2 image loss as the task-specific metric $\mathcal{L}_{\text{TASK}}$. Keeping in mind that the inner-workings of the original tonemapping operator are kept entirely opaque during proxy training *and* tonemapper hyperparameter optimization, we show first that both the tonemapper with the optimized tonemapping hyperparameters as well as the proxy itself reproduce the target output behavior (see Fig. 5). We obtain a PSNR of 44.31 dB on a test set of 19 manually tonemapped images.

5.2.2 Hyperparameter Optimization of Complex, Noisy, Non-linear Benchmark Functions. Next, we validate the proposed DPFs for general black-box non-linear optimization. To this end, we find the (apriori known) minima of several non-linear optimization benchmarking functions. Here, we employ 20-dimensional generalizations of four such functions, namely the Ackley [2012], Rastrigin [1974], Step-2 [Jamil and Yang 2013], and the Alpine-1 [Jamil and Yang 2013] functions (see the supplemental document for equations). We pollute each function with 0-mean Gaussian noise with a standard deviation equal to 3% of the maximum function value. The Ackley function is non-convex and multimodal; the Rastrigin function is convex, separable and multimodal; the Step-2 function is discontinuous, non-differentiable, separable and unimodal; and, the Alpine function is non-convex, multimodal and non-separable. All functions have a minimum output value of 0 at the origin. Our experiments confirm that differentiable proxies are a powerful tool for general non-linear optimization problems (Fig. 6). Of note, we outperform state-of-the-art Bayesian optimizers typically used for hyperparameter search in machine learning applications [Snoek et al. 2012].

6 HARDWARE ISP EXPERIMENTS

We will apply our differentiable proxy framework to black-box *hardware* ISP optimization.

6.1 Hardware ISP with RAW Injection

We first experiment on the ARM Mali-C71 ISP, a state-of-the-art hardware ISP with the ability to *inject* pre-captured RAW images to the ISP, bypassing optics and sensor. This allows us to directly evaluate f_{PROXY} function outputs for any input RAW image $\mathbf{I}_{\text{RAW}}$ and hyperparameter setting $\mathcal{P}$.

Here, we slightly modify the our DPF UNet architecture (Sec. 4.3) to expect RAW image patch inputs, i.e., untiled into four different

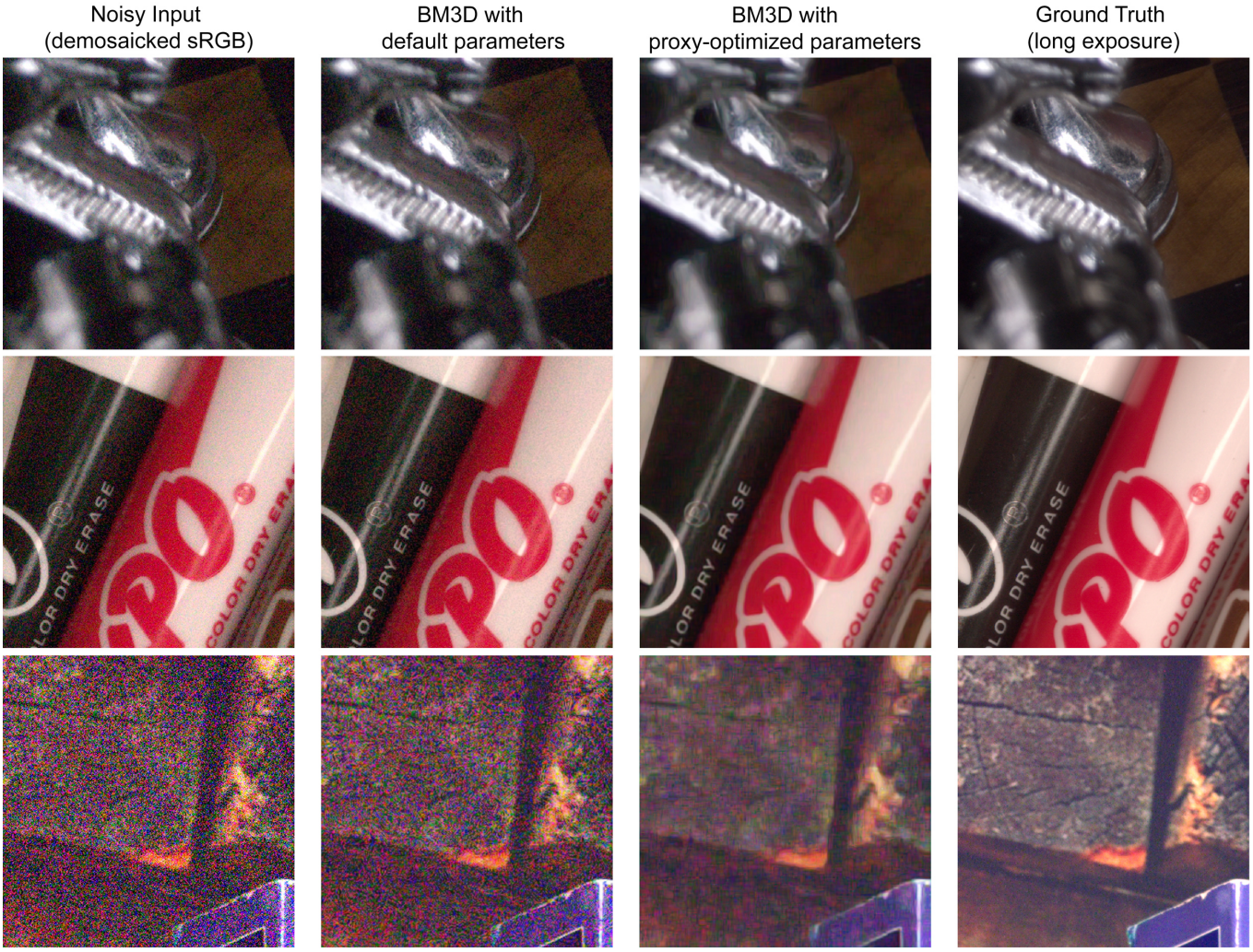


Fig. 4. Low-light smartphone denoising with proxy optimization. We illustrate sRGB-to-sRGB denoising on the SIDD sRGB dataset [Abdelhamed et al. 2018], consisting of measured smartphone images. By simply optimizing the hyperparameters of an established ISP, such as BM3D, we can improve its performance substantially. In fact, here, we outperform state-of-the-art methods (see Table 2, and the supplemental material for additional qualitative results).

channels (corresponding to R-G-G-B; as in [Chen et al. 2018]). With a 512×512 input image patch $\mathbf{I}$, the actual CNN input is therefore $256 \times 256 \times 4$. We shuffle the $256 \times 256 \times 12$ UNet output according to the sub-pixel layer described in [Shi et al. 2016] to reconstruct a three-channel 512×512 RGB output patch $\mathbf{O}_{\text{ISP}}$.

Similarly to our sRGB denoising experiment (Sec. 5.1), we optimize the hyperparameters of the ARM hardware ISP for a RAW denoising task on the SIDD dataset [Abdelhamed et al. 2018]. In a first stage (Sec. 4.2.1) we select 5×10^3 random 512×512 image patches from the RAW SIDD dataset, sampling random hyperparameters and injecting the RAW image patches directly to the ARM ISP to recover corresponding output images. With these input/output images, we train the DPF weights (Eq. 3).

In a second stage (Sec. 4.2.2), we fix the proxy weights and back-propagate the denoising loss back to the input hyperparameters. Here, we rely on the noisy image patches $\mathbf{I}_{\text{NOISY}}$ and their ground truth denoised counterparts $\mathbf{I}_{\text{CLEAN}}$ from the SIDD RAW training dataset. We obtain optimized ISP hyperparameters by solving Eq. 5, except now with a task-specific loss $\mathcal{L}_{\text{TASK}} = \mathcal{L}_{\text{PERC}} + \lambda\mathcal{L}_1$, where $\mathcal{L}_{\text{PERC}}$ is a *perceptual* loss (we use the AlexNet variant [Zhang et al. 2018]), $\mathcal{L}_1$ is a per-pixel L1 loss, and we use scalar weight $\lambda = 2$ in all our experiments to balance between the two losses.

Fig. 7 qualitatively compares the ISP's output generated default parameters and our proxy-optimized parameters. Images generated using our approach are higher quality, exhibit less noise, and more closely resemble the ground truth. Unfortunately, RAW injection capabilities are exceedingly rare as the vast majority of hardware ISPs do not support this feature and so, in these practical scenarios, f_{PROXY} cannot be directly evaluated. As such, we require another solution to support a broader set of modern hardware ISPs.

6.2 Prototype for Generic Black-box Hardware ISPs

To optimize hyperparameters for these more general hardware ISPs, we design a novel hardware prototype that directly measures ISP

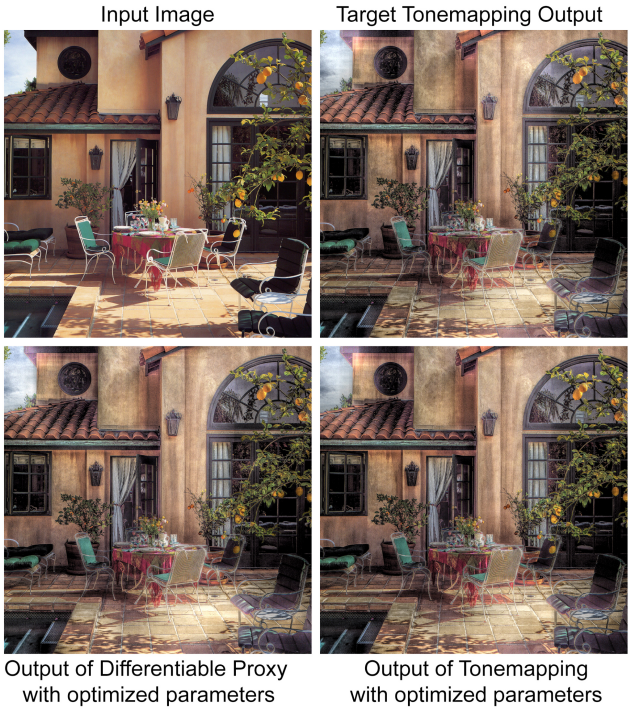


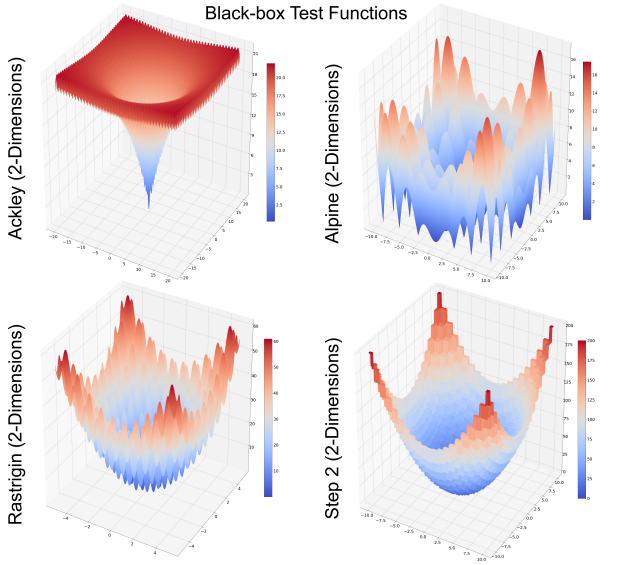
Fig. 5. Qualitative example of global image tonemapping, where a differentiable proxy is trained to accurately fit the tonemapping operator of Paris et al. [2011]. The proposed proxy models can accurately learn global image operations, beyond local, stencil-based image processing operations such as demosaicking or unsharp masking.

output resulting from captured stimuli: Fig. 9 summarizes our hardware setup, which consists of an LCD monitor to display stimuli images **I**, a camera calibrated to image these stimuli, an ISP that processes each capture with programmable ISP hyperparameters, and storage to recover the corresponding ISP output image $\mathbf{O}_{\text{ISP}}$. We use a Sony IMX290 capture board with a 10-bit interface and a fixed exposure time of 20 ms. We auto-select and fix the analog gain once, according to the lighting conditions. The camera board is equipped with a Sunex PN DSL219 lens. Our capture setup is compact, flexible, and fast, allowing us to automatically generate of large training dataset of images. We will further describe its calibration that allows us to obtain images $\mathbf{O}_{\text{ISP}}$ with pixel accurate alignment with **I**, before discussing how we generate training data. After which, we apply our hardware prototype to two applications: improving human-consumable image quality and automated object detection.

6.2.1 Prototype Calibration. We find accurate geometric correspondence between the output $\mathbf{O}_{\text{ISP}}$ and displayed **I** images by displaying a Gray code sequence[2] [Gupta et al. 2011] that allows us to calibrate a full light transport mapping matrix between camera and display pixels [Peers et al. 2009; Sen et al. 2005]. We keep the ISP tonemapping close to linear during this calibration process.

We position the monitor so that every one of its pixels is smaller than a sensor pixel in the output image. We assume a single iMac

[2]In practice, we employ green-and-black codes, since green is generally best resolved by Bayer sensors, minimizing the impact of chromatic aberration.



Test Functions (20D variants)	Differentiable Proxies	BayesOpt [2014]	HyperOpt [2013]	Powell [1965]	Nelder-Mead [1965]
Ackley	**16.84**	21.65	20.27	20.95	21.18
Rastrigin	**94.35**	124.91	137.79	209.42	258.51
Step-2	**6359.0**	14018.0	22444.0	49120.0	51669.0
Alpine	**24.48**	24.96	29.02	50.85	24.48

Fig. 6. Optimization performance benchmark—using differentiable proxies during optimization (i.e., with Adam) compares well to state-of-the-art 0^{th}-order methods on a number of well-establish benchmark functions. We visualize 2D variants of these test functions, but perform our test on 20D instances. The table reports loss function values after optimization. The best competing method, "BayesOpt", runs at around 10 minutes for these simple functions, while the proposed proxy model requires only a few seconds. This validates that simplex methods such as used in [Nishimura et al. 2018] do not scale to complex hyperparameter optimization problems.

5K monitor sufficiently covers the camera's field of view but, if not, multiple displays can trivially be used (and independently calibrated using the aforementioned procedure). We then resample the displayed target using the inverse of the calibrated light-transport matrix, using linear interpolation. Furthermore, we color-calibrate the monitor to emulate different illuminants (e.g., A, D65 and TL84). We validate hardware results (under different illuminants) and demonstrate that this calibration holds up to real-world capture scenarios.

6.2.2 Training Data Generation. To train a DPF to approximate a hardware ISP (Sec. 4.2.1) with our prototype, we acquire 1080×1920 RAW images of a specially-designed rainbow chart (see Fig. 1a). We designed the chart using random ellipse chart components for full reference image difference metrics. Using ellipses instead of discs, as is typical with Dead Leaves/Spilled Coins charts, ensures that the shape family is robust to lens distortion. It also ensures that a significant number of long/thin features are present in output images. To prevent demosaicking, denoising and sharpening thresholds due to overfitting of fixed contrast, we include a constrast gradient in the random ellipses chart: at one end, color values of nearby ellipses are close to each other (low contrast), and at the other end they are farther (high contrast). We also use hyperbolic wedge blocks modified

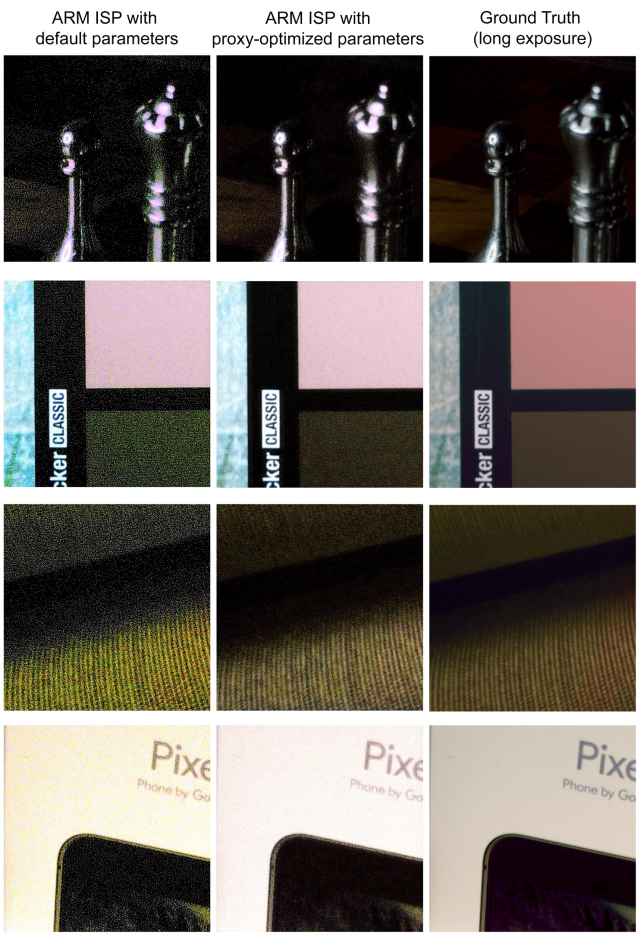


Fig. 7. Denoising results on the RAW SIDD dataset with an ARM Mali-C71 hardware ISP. We leverage RAW injection capabilities of this ISP to directly evaluate its performance on input RAW images, bypassing the optics and sensor. Compared to default ISP parameters (left), our approach (middle) finds hyperparameters that significantly improve visual quality compared to ground truth (right). See the supplemental document for additional results.

so that each frequency fills the width of the block. We lay out color and grey patches in two concentric "rainbows". In addition to minimizing the impact of lens shading, this configuration leads to patch interfaces at a variety of angles with respect to sensor lines. This minimizes horizontal and vertical bias tuning losses that measure interface cleanliness. Moreover, when used to quantify color accuracy, we compare output image patch averages to chromatically-adapted values derived from direct photospectrometer measurements. We insert skin gradients (or gradients between somewhat similar colors) in the chart to allow us to measure structured noise on patches that are smooth but not "flat", thus mitigating the tendency of some ISPs to posterize. Smooth gradients are generally included where lens shading is strong, i.e., near the edges of the field of view.

6.2.3 Modeling a Generic Hardware ISP with a Differentiable Proxy. We apply the first stage of our approach (Sec. 4.2.1) to generic hardware ISPs (that cannot accept direct RAW injection, unlike Sec. 6.1), we use the aforementioned prototype system to display our rainbow

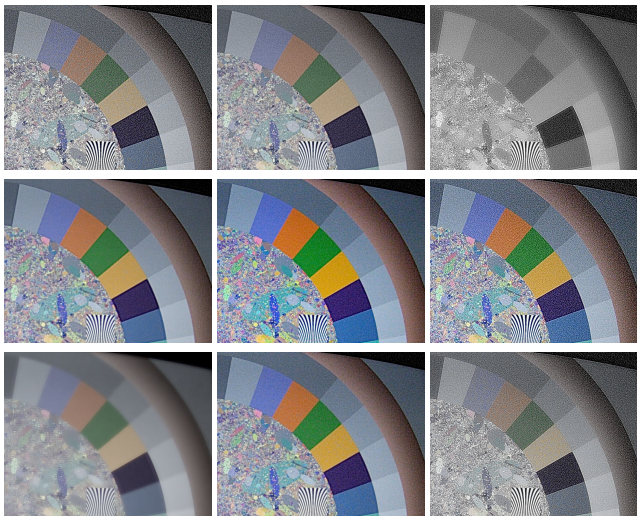


Fig. 8. Example sample image crops $\mathbf{O}_{\text{ISP}}$ obtained by imaging the same rainbow chart with an ISP while varying its hyperparameters in the first stage of the proxy training. A larger, randomly sampled training dataset is recorded as the training set for the initial proxy model fitting.

chart. Again, we randomly sample $M = 5 \times 10^3$ ISP hyperparameter settings $\mathcal{P}$ and capture the corresponding post-ISP output sRGB images $\mathbf{O}_{\text{ISP}}$. We also store the corresponding (resampled) ground truth chart image $\mathbf{I}_{\text{CHART}}$ before training the UNet CNN weights (Sec. 4.3; Eq. 3). Fig. 8 illustrates example images $\mathbf{O}_{\text{ISP}}$. Note that, while the proxy is only trained on the rainbow chart, it does not overfit to this particular image and generalizes well to other real images, as we demonstrate in the next section.

6.3 Applications for Optimized Black-box Hardware ISPs

We use our trained DPFs to optimize hyperparameters of a black-box ARM Mali-C71 hardware ISP for two tasks: perceptual quality improvement and object detection. We detail the second stage of our approach (Sec. 4.2.2) for these applications, below.

6.3.1 Improving Perceptual Image Quality. We optimize hyperparameters to improve perceptual quality of the ISP output. We first train a DPF f_{PROXY} for the ARM Mali-C71 hardware ISP with our display prototype (Sec. 6.2): as in Sec. 6.1, we use a sum of L1 and perceptual losses [Zhang et al. 2018]. Here, however, we compute the loss on the (resampled) rainbow chart (Sec. 6.2.2) and the ISP output image produced. Results, illustrated in Fig. 1 and 10, demonstrate visually-pleasing, high-quality images now produced by the hyperparameter-optimized ISP. Table 3 validates theses results quantitatively on perceptual and traditional image metrics. We generate these results automatically in *less than three hours* and they *improve* IQ compared to a *three-month* manual tuning by imaging experts.

We can also apply our method to modify the appearance of ISP output images in a deliberate manner: Fig. 11 illustrates an example where an end-user can force the proxy to learn purposefully over-sharpened, high-contrast images. Since no specific ISP hyperparameters directly control sharpening, obtaining such a tailored result by manual ISP tuning would be challenging. With our method,

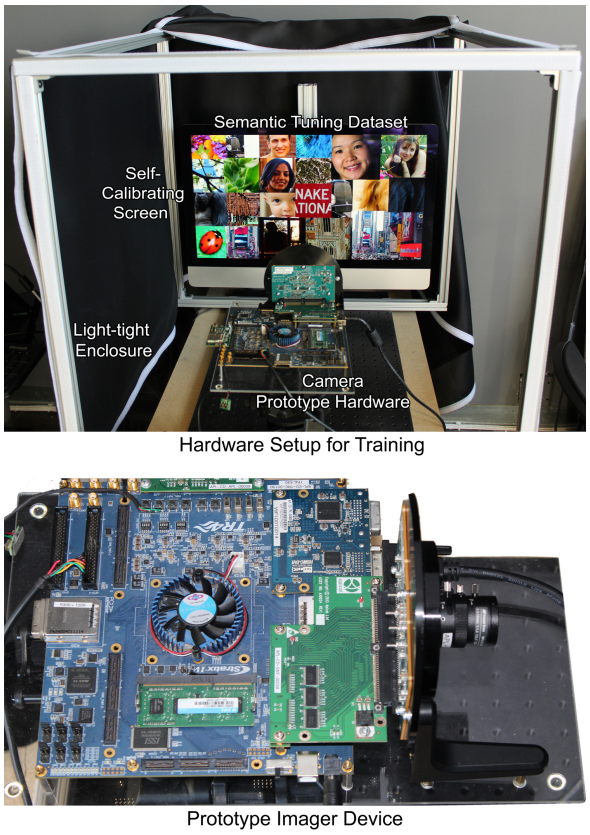


Fig. 9. Our *hardware-in-the-loop* system for optimizing black-box hardware ISPs. We image synthetic chart images (for the first stage proxy training), and task-specific images (for second stage optimization) on a display, which is placed in front of the camera. The mapping from display to camera pixels is calibrated accurately using a full light-transport calibration. We use a hardware capture board with Sony IMX 290 sensor and next-generation ARM Mali C71 hardware prototype ISP, running on an FPGA.

an end-user need simply modify the rainbow chart to the desired look in order to yield a proxy that replicates this effect.

6.3.2 2D Object Detection Task. We also apply our method to optimizing the ISP output in order to improve object detection results. Here, after training a DPF f_{PROXY} for the ARM Mali-C71 hardware ISP as above (Sec. 6.2), we additionally train an object detection network f_{FRCNN} to detect object bounding boxes on the KITTI dataset [Geiger et al. 2013]. We employ the commonly-used Faster R-CNN model [Ren et al. 2015] with a Resnet-101 backbone [He et al. 2016] (pretrained on ImageNet [Deng et al. 2009]), and fine-tuned on the KITTI training dataset. Our detector obtains 91.3% on a 20/80 KITTI test/training set split (measured for cars only; see [Geiger et al. 2013]). We can therefore consider this object detector as state-of-the-art. The f_{FRCNN} outputs bounding boxes and associated detection scores, and we train it by maximizing a typical *intersection-over-union* (IoU) object detection loss $\mathcal{L}_{\text{IOU}}$ on the KITTI dataset [Ren et al. 2015]. To obtain a DPF for the end-to-end object detection pipeline, we *chain* the two networks to obtain $f_{\text{DETECT}} = f_{\text{FRCNN}}(f_{\text{PROXY}})$. In other words, the ISP output image is fed directly to the Faster R-CNN object detector.

Fig. 10. Low-noise imaging example. This example demonstrates that the current approach achieves golden-eye tuned parameters which required months of manual optimization. The bottom inset examples visualize the improvement in color moire, measured in Table. 3.

Our second stage (Sec. 4.2.2), for this application, relies on the chained proxy f_{DETECT} when optimizing ISP hyperparameters. Here, DPF *and* Faster R-CNN weights are both fixed during hyperparameter optimization. We acquire 50 training images from an autonomous test vehicle in populated urban scenes (including cars, trucks, pedestrians, etc.) From these, we extract 50 low-resolution patches per image, and display them to the ISP with our display prototype. We obtain the ISP hyperparameters by back-propagating the IoU loss $\mathcal{L}_{\text{IOU}}$ computed between detections made by f_{DETECT} and the ground truth annotations.

To evaluate our approach, we acquire another non-overlapping set of 1068 RAW images using our hardware prototype system. Each object class is also hand-labelled (we use an external labeling service to avoid any bias), and we evaluate a total of 10590 ground truth objects. To avoid overfitting to our specific display prototype, we do not display these images to the ISP and instead adopt another strategy. As in Sec. 6.1, we record the test images in RAW (using the car camera) and *inject* the RAW images directly to the ISP, identical to the ISP used in real-world conditions.

Table 4 summarizes our quantitative tests, demonstrating a significant performance increase of 0.31 to 0.37 in mean average precision (50–90 mAP according to COCO metrics [Lin et al. 2014]) *with the same Faster R-CNN network*, when the ISP is optimized for object detection. As in Fig. 12, and as opposed to manual expert-tuned parameters, the optimized ISP does not output images suitable for human consumption (note the pink hue), but images that are rather preferred by the Faster R-CNN network. Of note, the solver learns to slightly blur objects, reducing noise, but also substantially improving bounding-box tightness and inter-class uncertainties.

7 DISCUSSION AND CONCLUSION

Limitations. Our approach has two notable limitations. First, our hardware-in-the-loop system is limited by the display's dynamic range. The range of real-world luminance cannot therefore be accurately reproduced. While we validate that our method generalizes to real scenes despite display gamut and dynamic range limitations (Sec. 6.3), we ignore distance-dependent aberrations and sensor dynamic range edge regions with the proposed setup. As such, HDR tasks require RAW-injection as described in Sec. 6.1. We could potentially mitigate this with recent, high dynamic range displays. Second, even though DPFs can apply to non-imaging optimization tasks (see Sec. 5.2.2), we rely on the convolutional nature of images in our image-based proxy models. Prior work shows that (even untrained) CNNs can serve as powerful image priors [Ulyanov et al. 2017], and so unnatural images (e.g., noise) may challenge our approach.

Table 3. Quantitative evaluation on Fig. 10 and at 16× gain. Results of the hardware ISP output after tuning are tested on the perceptual loss [Zhang et al. 2018] and traditional image quality metrics (lower is better), see Supplemental Material for details. Our method starts from a random input and improves on the human-tuned parameters.

	ARM Mali ISP (1× gain)		ARM Mali ISP (16× gain)	
	Manual Tuning	Proxy Optimized	Manual Tuning	Proxy Optimized
Perceptual Loss	0.244	**0.217**	0.523	**0.403**
Detail Accuracy	14.94	**13.03**	19.56	**18.79**
Color Accuracy	10.49	**10.23**	12.50	**12.36**
Zipper	0.111	**0.096**	0.227	**0.200**
Color Moire	2676	**727**	2836	**888**

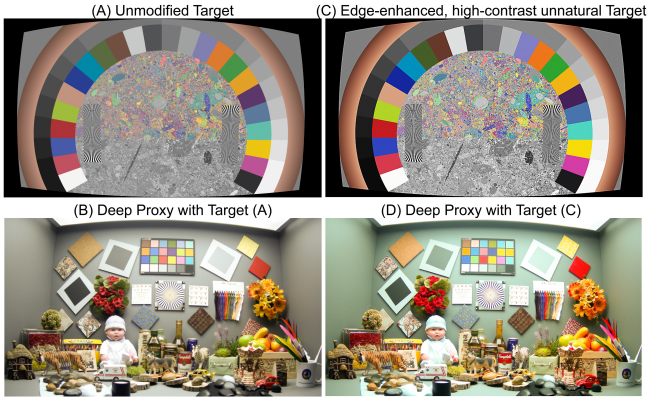


Fig. 11. Customized functions in our second stage: we oversharpen and contrast-boost the target, resulting in an *unnatural* target that the user can adjust.

Conclusion. We present a fully automatic approach for optimizing black-box ISPs, relying on *differentiable proxy functions* to model the parameterized ISP behavior. We *learn* to reproduce the totality of transformations an ISP applies to input images, as a function of the IPS's input hyperparameters. We parameterize DPFs with a well-established UNet CNN architecture. For black-box hardware ISPs, DPF training relies on a novel *hardware-in-the-loop* system that displays input images to the ISP with a calibrated high-resolution monitor. In this way we generate images suitable for proxy training in a supervised learning fashion. Once trained, we freeze proxy weights and backpropagate arbitrary task-specific loss through to the input ISP hyperparameters with standard 1$^{\text{st}}$-order methods.

We demonstrate the utility of DPFs on several domain-specific applications, broadly organized in two categories. First, we validate our approach on two software-based ISPs and four complex, non-linear optimization benchmarks; here, we outperform recent state-of-the-art image processing methods based on deep learning for extreme low-light denoising, automatically determining a more optimal set of parameters to apply to existing models. Second, we apply our method on hardware ISPs, where we test two tasks: optimizing the human-consumable outputs of an imaging system according to tailored perceptual metrics, and optimizing an imaging system for consumption by an object detector used in autonomous driving prototypes. Our more controllable calibration and testing framework generate images better suited to the application-specific

Table 4. Quantitative object detection results. By optimizing images for consumption by a pretrained object detection network, our approach ("Optimized") yields improved mean average precision (mAP) scores compared to a hand-tuned ISP ("Hand-tuned") on a challenging set of test images. Corresponding qualitative results are shown in Fig. 12.

Method	car/ van/ suv	bus/ truck/ tram	person	True positives	False negatives	mAP @[.5,.9] (COCO)
Hand-tuned	4101	97	662	4860	778	0.31
Optimized	4917	134	713	5764	587	0.37

Fig. 12. Qualitative object detection results. While the images produced by an ISP with expert-tuned parameters (left) may look more natural to the human eye, the performance of a pretrained object detector can be improved by adjusting the ISP parameters to maximize the intersection-over-union score (IoU) of the detector (right). The middle column shows the detection results with the initial tuning parameters.

optimization task, improving false positive rates and IoU on challenging low-light scenes. In the future, it would be interesting to explore how the estimation of dynamic control parameters, e.g., for auto-exposure. We also envision to expand the proposed proxy model across optics, sensor and detector hyperparameters, as a step towards a fully automated "evolution" of tomorrow's cameras.

REFERENCES

Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. 2018. A High-Quality Denoising Dataset for Smartphone Cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1692–1700.

D. Ackley. 2012. *A Connectionist Machine for Genetic Hillclimbing*. Springer US.

Michal Aharon, Michael Elad, Alfred Bruckstein, et al. 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing* 54, 11 (2006), 4311.

Brendan Barry, Cormac Brick, Fergal Connor, David Donohoe, David Moloney, Richard Richmond, Martin O'Riordan, and Vasile Toma. 2015. Always-on vision processing unit for mobile applications. *IEEE Micro* 35, 2 (2015), 56–66.

Donald Baxter, Frederic Cao, Henrik Eliasson, and Jonathan Phillips. 2012. Development of the I3A CPIQ spatial metrics. *Proc.SPIE* 8293. https://doi.org/10.1117/12.905752

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.

James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *12th Python in Science Conference*. Citeseer, 13–20.

A. Buades, B. Coll, and J.-M. Morel. 2005. A non-local algorithm for image denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2. 60–65.

Harold Burger, Christian Schuler, and Stefan Harmeling. 2012. Image denoising: Can plain neural networks compete with BM3D?. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4 (July 2017).

C. Chen, Q. Chen, J. Xu, and V. Koltun. 2018. Learning to See in the Dark. *ArXiv e-prints* (May 2018). arXiv:1805.01934

Q. Chen, J. Xu, and V. Koltun. 2017. Fast Image Processing with Fully-Convolutional Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2516–2525. https://doi.org/10.1109/ICCV.2017.273

Yunjin Chen and Thomas Pock. 2017. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE transactions on pattern analysis and machine intelligence* 39, 6 (2017), 1256–1272.

J. Choi, S. Jang, S. Lee, Y. Hwang, and B. H. Choi. 2014. Memory optimization of bilateral filter and its hardware implementation. In *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*. 1–2. https://doi.org/10.1109/ISCE.2014.6884437

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. 2007. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Processing* 16, 8 (2007).

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. on Computer Vision and Pattern Recognition*. 248–255.

Michael Elad and Michal Aharon. 2006. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing* 15, 12 (2006), 3736–3745.

Qingnan Fan, Jiaolong Yang, David Wipf, Baoquan Chen, and Xin Tong. 2018. Image Smoothing via Unsupervised Learning. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA 2018)* 37, 6 (2018).

Kenneth Garrard, Thomas Bruegge, Jeff Hoffman, Thomas Dow, and Alex Sohn. 2005. Design tools for freeform optics. In *Current Developments in Lens Design and Optical Engineering VI*, Vol. 5874. International Society for Optics and Photonics, 58740A.

Carl Friedrich Gauss. 1843. *Dioptrische Untersuchungen von CF Gauss*. in der Dieterichschen Buchhandlung.

Joseph M Geary. 2002. *Introduction to lens design: with practical ZEMAX examples*. Willmann-Bell Richmond.

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

M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. 2016. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 191.

M. Gharbi, J. Chen, J. Barron, S. Hasinoff, and F. Durand. 2017. Deep Bilateral Learning for Real-Time Image Enhancement. *ACM Trans. Graph. (SIGGRAPH)* (2017).

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. 1998. NeuroAnimator: Fast Neural Network Emulation and Control of Physics-based Models. In *Proc. of the 25th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM.

Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. 2014. Weighted nuclear norm minimization with application to image denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. 2018. Toward convolutional blind denoising of real photographs. *arXiv preprint arXiv:1807.04686* (2018).

Mohit Gupta, Amit Agrawal, Ashok Veeraraghavan, and Srinivasa G Narasimhan. 2011. Structured light 3D scanning in the presence of global illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 713–720.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation* 11, 1 (2003), 1–18.

S. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. Barron, F. Kainz, J. Chen, and M. Levoy. 2016. Burst Photography for High Dynamic Range and Low-light Imaging on Mobile Cameras. *ACM Trans. Graph.* 35, 6, Article 192 (2016), 12 pages.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

F. Heide, M. Steinberger, Y.-T. Tsai, M. Rouf, D. Pajak, D. Reddy, O. Gallo, J. Liu, W. Heidrich, K. Egiazarian, J. Kautz, and K. Pulli. 2014. FlexISP: A flexible camera image processing framework. *ACM Trans. Graph. (SIGGRAPH Asia)* 33, 6 (2014).

ISO. [n. d.]a. ISO 1858. https://standards.ieee.org/standard/1858-2016.html. ([n. d.]). [Online; accessed 5-January-2019].

ISO. [n. d.]b. ISO 71696. https://www.iso.org/standard/71696.htm. ([n. d.]). [Online; accessed 5-January-2019].

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Momin Jamil and Xin-She Yang. 2013. A Literature Survey of Benchmark Functions For Global Optimization Problems. *CoRR* abs/1308.4008 (2013). arXiv:1308.4008 http://arxiv.org/abs/1308.4008

Norman Koren. 2006. The Imatest program: comparing cameras with different amounts of sharpening. In *Digital Photography II*, Vol. 6069. International Society for Optics and Photonics, 60690L.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6 (2018), 222:1–222:11.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. 2019. Beyond Pixel Norm-Balls: Parametric Adversaries using an Analytically Differentiable Renderer. In *International Conference on Learning Representations*.

Ilya Loshchilov, Tobias Glasmachers, and Hans-Georg Beyer. 2017. Limited-Memory Matrix Adaptation for Large Scale Black-box Optimization. *CoRR* abs/1705.06693 (2017). http://arxiv.org/abs/1705.06693

Daniel Malacara-Hernández and Zacarías Malacara-Hernández. 2016. *Handbook of optical design*. CRC Press.

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

ON Semi MT9P111. 2015. MT9P111: 1/4-Inch 5 Mp System-On-A-Chip (SOC) CMOS Digital Image Sensor. http://www.onsemi.com/pub/Collateral/MT9P111-D.PDF. (2015).

John A Nelder and Roger Mead. 1965. A simplex method for function minimization. *The computer journal* 7, 4 (1965), 308–313.

J. Nishimura, T. Gerasimow, R. Sushma, A. Sutic, C. Wu, and G. Michael. 2018. Automatic ISP Image Quality Tuning Using Nonlinear Optimization. In *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2471–2475. https://doi.org/10.1109/ICIP.2018.8451818

Sylvain Paris, Samuel W Hasinoff, and Jan Kautz. 2011. Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid. *ACM Trans. Graph.* 30, 4 (2011).

Pieter Peers, Dhruv K Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Debevec. 2009. Compressive light transport sensing. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 3.

Jonathan B. Phillips and Henrik Eliasson. 2018. *Camera Image Quality Benchmarking* (1st ed.). Wiley Publishing.

MJD Powell. 1965. A method for minimizing a sum of squares of non-linear functions without calculating derivatives. *Comput. J.* 7, 4 (1965), 303–307.

R. Ramanath, W. Snyder, Y. Yoo, and M. Drew. 2005. Color image processing pipeline in digital still cameras. *IEEE Signal Processing Magazine* 22, 1 (2005), 34–43.

L. A. Rastrigin. 1974. Systems of extremal control. *Nauka* (1974). https://ci.nii.ac.jp/naid/10018403158/en/

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

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention*.

Stefan Roth and Michael J Black. 2005. Fields of experts: A framework for learning image priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2.

Pradeep Sen, Billy Chen, Gaurav Garg, Stephen R Marschner, Mark Horowitz, Marc Levoy, and Hendrik Lensch. 2005. Dual photography. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 745–755.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2016. Taking the human out of the loop: A review of bayesian optimization. *IEEE* 104, 1 (2016), 148–175.

L. Shao, R. Yan, X. Li, and Y. Liu. 2014. From Heuristic Optimization to Dictionary Learning: A Review and Comprehensive Comparison of Image Denoising Algorithms. *IEEE Transactions on Cybernetics* 44, 7 (2014), 1001–1013.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Vincent Sitzmann, Steven Diamond, Yifan Peng, Xiong Dun, Stephen Boyd, Wolfgang Heidrich, Felix Heide, and Gordon Wetzstein. 2018. End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 114.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*.

R. Stead. 2016. P2020 - Standard for Automotive System Image Quality. https://standards.ieee.org/develop/project/2020.html. (2016).

David G Stork and Patrick R Gill. 2013. Lensless ultra-miniature CMOS computational imagers and sensors. (2013).

David G Stork and Patrick R Gill. 2014. Optical, mathematical, and computational foundations of lensless ultra-miniature diffractive imagers and sensors. *International Journal on Advances in Systems and Measurements* 7, 3 (2014), 4.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-task bayesian optimization. In *Advances in neural information processing systems*. 2004–2012.

Hossein Talebi and Peyman Milanfar. 2014. Global image denoising. *IEEE Trans. Image Process* 23, 2 (2014), 755–768.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*. IEEE, 839–846.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. Deep image prior. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Li Xu, Jimmy Ren, Qiong Yan, Renjie Liao, and Jiaya Jia. 2015. Deep Edge-Aware Filters. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 1669–1678. http://proceedings.mlr.press/v37/xub15.html

Hao Zhang, Wenjiang Liu, Ruolin Wang, Tao Liu, and Mengtian Rong. 2016. Hardware architecture design of block-matching and 3D-filtering denoising algorithm. *Journal of Shanghai Jiaotong University (Science)* 21, 2 (2016), 173–183.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing* 26, 7 (2017), 3142–3155.

Lei Zhang, Weisheng Dong, David Zhang, and Guangming Shi. 2010. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition* 43, 4 (2010), 1531–1549.

L. Zhang, X. Wu, A. Buades, and X. Li. 2011. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic Imaging* 20, 2 (2011).

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Daniel Zoran and Yair Weiss. 2011. From Learning Models of Natural Image Patches to Whole Image Restoration.