Auto-ISP: An Efficient Real-Time Automatic Hyperparameter Optimization Framework for ISP Hardware System

Abstract

Image Signal Processor (ISP) is widely used in intelligent edge devices across various scenarios. The intricate and time-consuming tuning process demands substantial expertise. Current AI-based auto-tuning operates discretely offline, relying on predefined scenes with human intervention, leading to inconvenient manipulation, with potentially fatal impacts on downstream tasks in unforeseen scenes. We propose a real-time automatic hyperparameter optimization ISP hardware system to address real-world scenarios. Our design features a tri-step framework and a hardware accelerator, demonstrating superior performance in human and computer vision tasks, even in real-time unforeseen scenes. Experiments showcase its practicality, achieving 1080P@75FPS/240FPS in FPGA/ASIC, respectively.

1 Introduction

The ever-expanding realm of applications on Intelligent Edge Devices (IEDs) significantly enhances people's lives, enabling seamless actions like sharing photos via smartphones, indulging in immersive experiences with AR/VR glasses, navigating using autonomous cars, and even utilizing drones for food delivery [18]. At the core of this capability lies perception, executed through Image Signal Processor (ISP) hardware, a crucial element that enables IEDs to interact with the real world. The ISP hardware plays a vital role in converting photoelectric signals collected by various camera sensors into uniform standard RGB signals, a process essential for downstream tasks [12]. This conversion is accomplished through the integration of multiple modules, employing specific image processing algorithms pipelined within the ISP.

However, integrated modules alone do not ensure the harmonious operation of an ISP system. It's crucial to establish the appropriate parameters for each module so that they operate optimally. For instance, improper ISP parameters can result in visual distortion or a decline in downstream task performance, even leading to serious consequences, such as a self-driving car misinterpreting a stop sign. This process of parameter adjustment is referred to as ISP parameter tuning. Yet, this task is far from trivial. Firstly, an ISP comprises numerous modules, each with multiple parameters, and each parameter has a broad range of available values. Secondly, optimizing a single module is not sufficient; there's a need to harmonize each module simultaneously with the others.

Traditional tuning methods involve manual adjustments by specialized engineers known as "golden eyes." This **manual-tuning** approach is not only time-consuming and labor-intensive but is also reliant on the experience and subjective judgment of the tuning engineers. For instance, even the most experienced ISP tuning teams may require three to six months to identify an appropriate set of parameters for an ISP pipeline [15]. Worse, images and videos processed by ISPs are increasingly employed for the deep neural network (DNN) based downstream tasks, such as intelligent surveillance and autonomous driving. Manual-tuning proves unsuitable

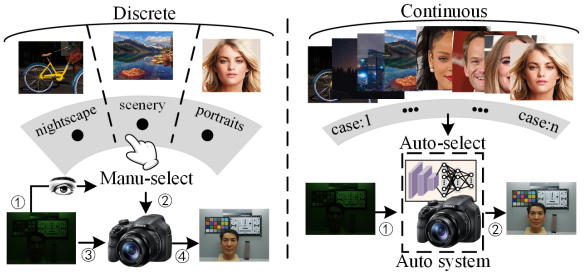

Figure 1: Our motivation example.

for such tasks since human experts do not possess the knowledge of which parameters are optimal for the machine vision [6].

A recent advancement in ISP tuning involves a proxy-based **auto-tuning** approach [15]. This method employs a DNN to emulate the ISP function and identifies optimal parameters through backpropagation. Remarkably, this technique can accomplish the parameter adjustment within a few days and is adaptable to various tasks. However, akin to manual-tuning, auto-tuning initiates by conducting an **offline search** for a set of parameters tailored to a specific scene. These parameters are then stored in the ISP hardware's memory. When the scene changes, the ISP necessitates reconfiguration based on the pre-stored parameters in memory.

Our motivation. The dependence on offline search poses a notable obstacle to the usability and scalability of auto-tuning. Despite the automated parameter search, its application still requires manual intervention. As depicted on the left side of Figure 1, it is essential to predefine a set of discrete scenes, such as portraits and nightscapes. In practical applications, manual selection of these predefined scenes becomes necessary for ISP reconfiguration. Furthermore, offline search typically learns from a limited set of predefined discrete scenes. Due to the vast parameter search space (see Table 1), it is difficult to efficiently approximate the learned parameters to unknown scenarios, which not only leads to significant degradation of ISP performance but also makes extensive continuous learning in real-world scenarios difficult to achieve. These challenges prompted us to investigate the automatic tuning of ISP parameters through online search.

In this paper, we introduce the architecture design and hardware implementation of a hyperparameter optimization framework tailored for real-time and continuous optimization of ISP parameters in real-world scenarios (refer to the right side of Figure 1). In particular, in the architecture design, we propose a **high-fidelity proxy** and a DNN-based **ParaNet** to improve the speed and accuracy of ISP parameter search. In the hardware implementation, we propose a **semi-parallel architecture** and **row-aware** processing techniques to align with the high throughput requirements of ISP hardware.

Our Auto-ISP, for the first time, successfully achieves real-time online parameter search and image processing. Our experimental results show our approach currently stands as the fastest ISP automatic parameter search, offering superior image quality compared

Table 1: Parameters (w/ range) for each module in our ISP.

Module	Parameter	Module	Parameter	Module	Parameter
DGAIN	$P_1 \in [0, 100]$	GB	$P_6 \in [200, 2000]$	DNS	$P_4 \in [60, 160]$ $P_5 \in [20, 102]$
DPC	$P_2 \in [0, 2047]$ $P_3 \in [0, 2047]$	WB	Fixed	EE	$P_7 \in [3, 6]$ $P_8 \in [0, 127]$
		DMC	None		
Total parameter space:		3.2746×10^{18}			

to previous search techniques, particularly in cases involving unforeseen changes. Our main contributions can be summarized as follows:

- We introduce a tri-step framework comprising a high fidelity proxy emulating an ISP, a ParaNet directly mapping images to parameters, and an online tuning module. This framework achieves real-time online searching of ISP parameters with both high accuracy and speed.
- We design an Auto-ISP hardware architecture seamlessly integrating a real-time automatic hyperparameter optimization framework. Our innovation includes a semi-parallel architecture and row-aware processing modules, aiming to reduce the resource gap between neural network acceleration and traditional ISPs.
- We systematically assess our design, yielding results that markedly outperform the State of the Art (SOTA). Furthermore, our implementation attains 75FPS and 240FPS at 1080P on FPGA and ASIC (28nm), respectively. This accomplishment serves as a robust verification of the performance and usability of our design.

2 Background

2.1 ISP Pipeline

The ISP pipeline is designed to convert raw data, captured from a camera sensor in the RGGB Bayer pattern, into a visually pleasing sRGB pattern [3]. This intricate process involves the application of various image processing algorithms, including Digital Gain, denoising [4], demosaicing [16], among others. Each of these algorithms is dedicated to handling a specific vision task in a carefully orchestrated, pipelined manner. Below we briefly describe some of the modules in the ISP pipeline that are important and used in our paper. **Digital Gain (DGain)** amplifies the amplitude of RAW data and limits the overflow. **Defeated Pixel Correction (DPC)** corrects bright or dark pixels in RAW data. **Denoise (DNS)** removes noise from RAW data to enhance clarity. **White Balance (WB)** balances channel colors by adjusting the gains of the 3 colors (RGB). **Green Balance (GB)** reduces differences in green pixels (Gr, Gb) in different rows. **Demosaic (DMC)** generates RGB images by interpolation of RAW data. **Edge Enhancement (EE)** sharpens RGB images affected by DNS or lens blur.

As listed in Table 1, the ISP operates within an immensely vast parameter space, where parameters are intricately interconnected. The RGB image, being the combined outcome of RAW data and all parameters, faces the challenge that optimizing the parameters of a single module may not yield an optimal effect for the entire ISP.

2.2 Automatic Offline Parameter Searching

We categorize previous works on ISP hyperparameter optimization into two distinct groups: non-proxy-based and proxy-based.

Non-proxy-based. [13] proposes a search framework utilizing evolutionary algorithms is introduced for joint optimization involving ISP and object detection. [10] proposes a mixed 0^{th}-order and 1^{th}-order optimizer to jointly optimize ISP modules and object detection. [14] presents an end-to-end camera design approach that concurrently optimizes compound optics with hardware and software ISP. [6] proposes a black-box searching approach based on CMA-ES strategy to reduce the parameter search space, and this method can directly integrate hardware into the search framework for parameter discovery.

Proxy-based. Proxy-based optimizations consist of two types: black-box optimization and white-box optimization. The distinction depends on whether the algorithm is transparent to the tuner. Black-box approach [15] employs a differentiable proxy function for parameter optimization and obtains specialized parameters for different tasks. This work marks the first instance of an automatic parameter optimization method based on a black-box system. White-box approach [17] introduces a reconfigurable ISP module search method for low-light denoising and object detection within pipeline optimization. **The above works with offline search are iterative and approach the optimal parameters slowly, rendering them less responsive to unforeseen scenario changes in the real world.**

3 Our Approach

Our approach leverages DNNs to automatically estimate and select an appropriate set of ISP hyperparameters in real-world scenes. The idea is to design a tri-step framework: 1) Mimic the non-differentiable ISP hardware; 2) Map the raw image to ISP hyperparameter; 3) Online tuning.

3.1 Tri-step Framework

Step 1–Mimic the non-differentiable ISP hardware. The discrete nature of hyperparameters in ISP hardware poses a challenge for optimization using DNNs, as it leads to a non-differentiable black-box function. This black-box function, essentially one that cannot be differentiated, presents a barrier to DNN optimization. To address this, we make the hardware ISP differentiable by training a DNN-based ISP proxy, following the approach outlined in [15].

As shown in 2(a), the paired training dataset is compiled from the ISP hardware, with raw data and a configuration file serving as input, and the corresponding output of the ISP hardware serving as ground truth. For each raw image, a set of randomly generated configuration files is created, each containing hyperparameters for tuning. These configuration files are then fed into the ISP hardware, with the output considered as the ground truth for training the proxy. Our training process can be expressed as:

$$\mathcal{W}_p = \arg\min_{\mathcal{W}_p}(\mathcal{L}_2(f_{proxy}(I_{RAW}, P, \mathcal{W}_p), f_{ISP}(I_{RAW}, P))), \quad (1)$$

where $\mathcal{W}_p$ is the weights of the proxy, P is the parameters of the ISP, I_{RAW} is the RAW data, f_{proxy} is the DNN proxy, f_{ISP} refers to the ISP hardware, and $\mathcal{L}_2$ represents the L2 loss.

Step 2–Raw image to ISP hyperparameter mapping. The raw image contains a wealth of detailed information, although it may seem diluted. This low-level information can be effectively utilized to generate high-level interpretations of the image. For example, the mean value of a raw image can be used to determine the ISO information or illumination information. The standard variations present the complexity and the noise strength of this image. Therefore, instead of backpropagating (see Figure 2(a)) the

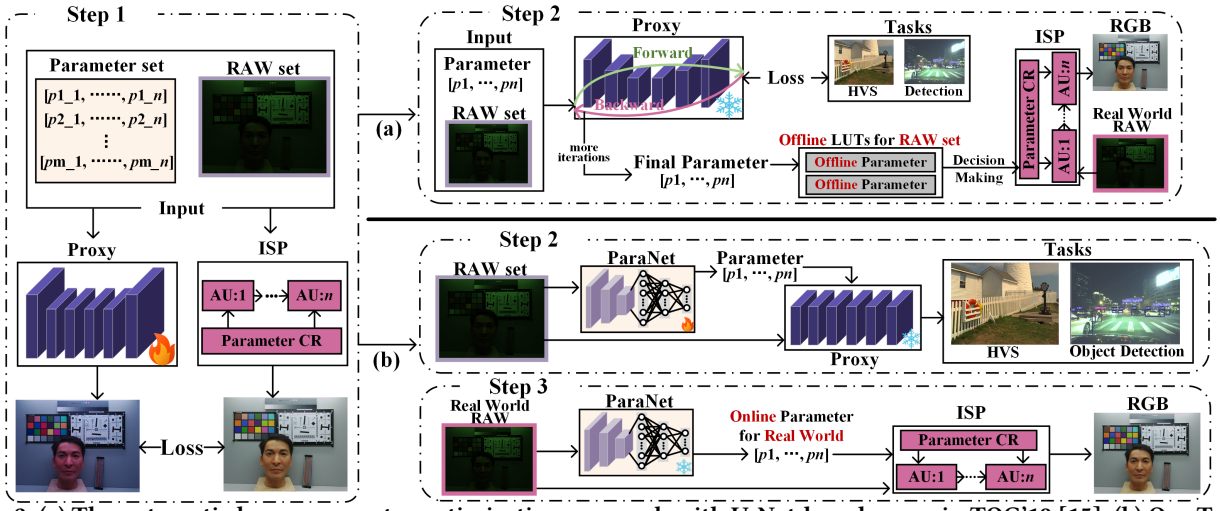

Figure 2: (a) The automatic hyperparameter optimization approach with U-Net-based proxy in TOG'19 [15]; (b) Our Tri-step automatic hyperparameter optimization approach with high fidelity proxy.

loss function to optimize each hyperparameter under specific predefined scenes, we employ a DNN-based network, termed **ParaNet**, to directly learn the raw to hyperparameter mapping process. In this way, the hyperparameter optimization process can be run online.

As illustrated in Figure 2(b), our design involves feeding the raw data into ParaNet to generate hyperparameters. These parameters, along with the raw data, are then processed through the frozen proxy ISP to calculate the loss specific to tasks (HV or CV). Subsequently, this loss is employed to update ParaNet. This process can be expressed as below with f_{para} and $\mathcal{W}_r$ representing ParaNet and its weights, respectively.

$$\mathcal{W}_r = \arg\min_{\mathcal{W}_r}(\mathcal{L}_{task}(f_{proxy}(I_{RAW}, f_{para}(I_{RAW}, \mathcal{W}_r), \mathcal{W}_p))), \tag{2}$$

Step 3–Online Tuning. As shown in Figure 2(b) step 3, we integrate ParaNet into the ISP hardware system, where raw data is fed into both ParaNet and ISP. Simultaneously, ParaNet is connected to ISP's configuration interface to export optimized parameters online. This process can be expressed as:

$$O_{rgb} = f_{ISP}(I_{RAW}, f_{para}(I_{RAW}, \mathcal{W}_r)), \tag{3}$$

where O_{rgb} is ISP's output RGB image.

3.2 A High Fidelity Proxy

The effectiveness of the proposed framework greatly depends on the accuracy of the proxy network in mimicking the ISP hardware. A more accurate proxy greatly facilitates the parameter optimization process. The proxy network in previous work was built based on the U-Net [15], due to its low computational cost and fast inference [11]. However, since the modules in the ISP pipeline are all low-level tasks, we find that the downsampling process in U-Net may discard such low-level details, resulting in inaccurate proxy. Therefore, we design the High Fidelity Proxy (HFP) using Residual-in-Residual Dense block (RRDB) [19] since it has no downsampling operations.

HFP consists of 2 RRDBs (f_{RRDB}) with skip connections. Each RRDB employs 6 RRDB units and 1 Conv2d unit. The raw data, as the input of HFP, were divided into four channels based on the color channels of the Bayer array in our design. In addition to the raw data, we concatenate as many channels as there are hyperparameters P in the ISP, where each channel is simply the value of the hyperparameter replicated over the spatial dimension. Further, the parameters also concatenate with the intermediate feature maps of each RRDB to strengthen parameter impacts. This process can be expressed as follows:

$$O_{f1} = f_{RRDB}(f_C(f_{down}(I_{RAW}), P))) + f_C(f_{down}(I_{RAW}), P), \tag{4}$$

$$O_{f2} = f_{RRDB}(f_C(O_{f1}, P))) + f_C(O_{f1}, P), \tag{5}$$

$$O_{RGB} = f_C(f_{up}((f_{up}(O_{f2})))). \tag{6}$$

4 Auto-ISP Hardware System

4.1 Dual-Pipeline Hardware Architecture

Our hardware architecture consists of two parts: ParaNet and ISP. DNN hardware incurs significant resource costs for low latency and high performance. However, in Auto-ISP, due to the sensor's raster scanning [5], the sensor outputs one pixel per cycle instead of a patch. The sampling speed of the sensor becomes the computational bottleneck for the entire system, rather than the ParaNet or ISP. Therefore, we propose a hardware-friendly dual-pipeline architecture to reduce hardware resources while ensuring no additional latency. The ParaNet pipeline and ISP pipeline execute in parallel to achieve real-time processing, as shown in Figure 3.

ParaNet hardware design. Basicly, a convolution module contains input channels (c_{in}) × output channels (c_{out}) convolution kernels. The high-parallel computing units result in significant resource utilization. Since the input of ParaNet is downsampled RAW data, reading adjacent 2 pixels every 8 cycles, we can take advantage of the downsampling interval to serialize the convolution computation for output channels. We proposed a **semi-parallel** architecture, a method that parallelizes input channels while serializing the computation for output channels, to reduce the 3x3 Conv kernel by a factor of c_{out}. In the ParaNet architecture, all layers, including Conv layers and fully connected layers, are implemented in semi-parallel manner. Due to the sensor readout scheme, the serialized output design can save large hardware resources without impacting the system latency.

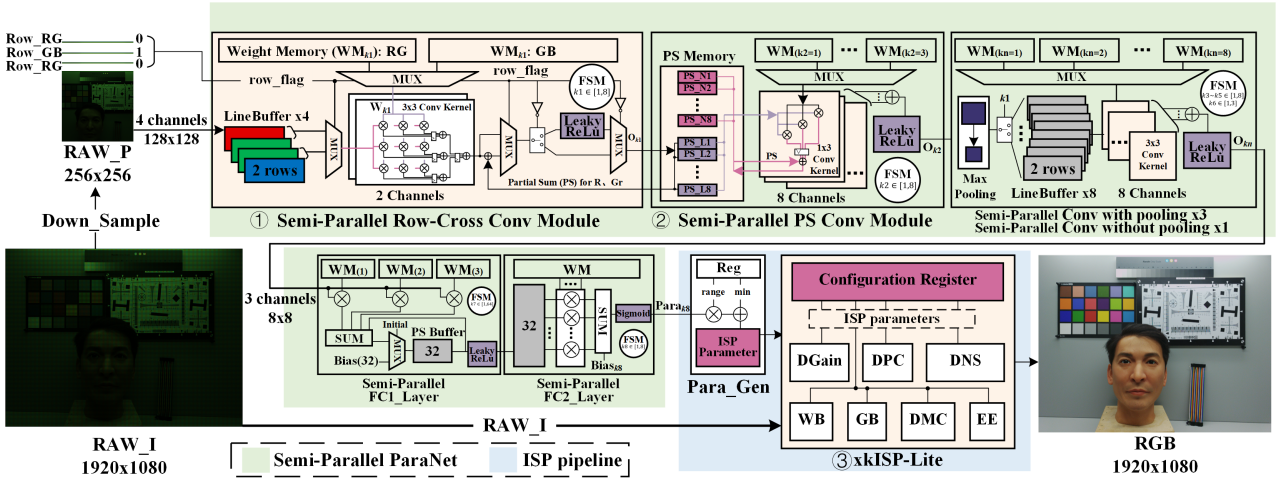

Figure 3: Auto-ISP: real-time ISP hardware architecture with online automatic hyperparameter optimization.

ISP pipeline. The output of the ParaNet will first be denormalized by the Para_Gen module and temporarily stored in the configuration register. The ISP hardware, as shown in Figure 3 ③, then reads this register to configure each image processing module in ISP pipeline. Note that ParaNet and ISP pipelines are processed in parallel with the optimized hyperparameters generated from the previous frame and used in the current frame.

Uniqueness. Unlike general patch-based neural network architectures, we leverage the sensor's sampling bottleneck to reduce a significant number of Conv kernels and fully connected layers' multiplier units without additional latency. We employ parallel computation in dual pipelines while ensuring that the computational latency of ParaNet and the register configuration delay do not exceed the processing latency of ISP, enabling frame-by-frame searching.

4.2 Row-Aware Processing Modules

In raster scanning, odd (contains R and Gr color pixels) and even rows (contains Gb and B) are alternately output from the sensor. Since odd and even rows respectively contain half of the input channels, general Conv kernels need to wait for the data from even rows to satisfy the patch size in full channels. Due to the odd-even row difference of raster scanning, we propose two types of row-aware processing modules for the first two Conv layers, that is the first Conv layer is implemented as a Row-Cross Conv module, and the second layer is the Partial Sum (PS) Conv module. The following will demonstrate how this design can further minimize hardware resources

Row-Cross Conv module. Odd and even rows contain different color pixels, resulting in a time difference in reading different color pixels. As shown in Figure 3 ①, we time-multiplex the 3x3 Conv kernel based on the time difference in reading odd and even rows to reduce the number of kernels by half in the Row-Cross Conv module. When reading odd rows, our Row-Cross Conv module calculates the PS of R and Gr channel convolutions, and when reading even rows, we accumulate the convolution result for Gb and B channels with PS. Building upon the semi-parallel architecture, the Row-Cross Conv module reduces the number of convolution kernels from 4 to 2 without introducing additional delay.

PS Conv module. Unlike the last 4 Conv modules, the memory usage of the 1st and 2nd Conv modules is huge due to the large size of RAW_P. Besides that, the max pooling is not applied in the first two Conv layers, aggravating memory capacity usage. As shown in Figure 3 ②, by alternately accumulating and overwriting, PS Conv with a stride of 2 reduces the linebuffers (PS_N) in PS Memory from 2 rows to 1 row. We also split the 3×3 Conv kernel into multiple batches of 1x3 PS Conv to reduce the computation units in Conv kernel.

Uniquesness. Based on the semi-parallel architecture, which takes advantage of downsampling delays, we further exploit the inherent difference in color between odd and even rows in sensor raster scanning. We introduce the Row-Cross Conv module and PS Conv module to reduce the number of Conv kernels and memory without introducing additional delay.

5 Evaluation

5.1 Experimental Setup

We conduct a comprehensive comparison of our design with both manual- and AI- tuning. For the manual tuning baselines, we choose default ISP parameters and those tuned by five ISP experts over four-hour. As the AI-tuned baseline, we select the SOTA TOG'19 [15].

Datasets and downstream tasks. We create a dataset of 30K sets of RAW-Parameter-RGB pairs for evaluating parameter tuning, using the open-source *xkISP*-Lite [7]. All raw images are sourced from the Sony IMX485 sensor, and configuration files with hyperparameters for tuning are randomly generated. We also evaluate our design using several downstream tasks. For training and testing in the human vision task, we utilize the Pixelshift200 [8] and Kodak datasets. The RAW data in Kodak dataset is generated following the approach outlined in [1]. For the CV task (i.e., object detection for nighttime autonomous car), we use the OnePlus dataset [17]. The object detection model utilized is YOLO-v3 [9]. For evaluating the robustness of our design in unforeseen scenes, we use SID dataset [2] for nighttime HVS tasks.

Training and metric. Our training takes place on the NVIDIA-A6000. The proxy network undergoes training for 2M iterations using L_2 loss and a learning rate of 10^{-5}. ParaNet undergoes 20K training iterations with a learning rate of 10^{-4}, also utilizing L_2

Figure 4: The visual comparison for human vision.

Figure 5: The visual comparison for object detection.

Table 2: The objective and perceptual quality of HVS.

Work	LPIPS↓	PSNR↑	SSIM↑	NIQE↓	Time↓
Default	0.3588	13.1071	0.6779	9.3315	-
Experts	0.3520	14.0436	0.6925	5.9799	4h
TOG'19 [15]	0.3253	16.2256	0.7837	7.0710	1.561s
Ours (U-Net proxy)	0.3217	16.0153	0.7675	6.3353	**3.255ms**
Ours (Our HFP)	**0.3005**	**17.6885**	**0.8154**	**5.8983**	

loss. We use PSNR and SSIM for an objective evaluation, LPIPS and NIQE for a subjective evaluation, and mAP for the detection task.

5.2 Performance

Human vision tasks. As listed in Table 2, our Tri-step framework with HFP achieves the highest scores on both objective and subjective metrics, notably yielding a significantly higher PSNR (1.4629dB) compared to the SOTA. A comparison between the U-Net proxy and HFP in our design demonstrates the effectiveness of the proposed HFP, showing a 1.6732dB improvement in PSNR. It also reveals that the proposed framework positively impacts subjective scores, such as LPIPS and NIQE, while the proxy architecture improvement contributes more to objective metrics like PSNR and SSIM. Moreover, our inference time of only 3.255ms, significantly outperforms the SOTA (1.561s). Visual results in Figure 4 illustrate that images processed by the ISP using our approach closely resemble the Groundtruth in terms of both brightness and detail.

Object detection tasks. Table 3 presents compelling results that underscore the effectiveness of our framework. Our approach achieves an mAP of 0.5001, significantly surpassing the SOTA's 0.3718. In Figure 5, it is evident that our approach excels in detecting more objects. A counterintuitive observation arises in nighttime driving–brighter does not necessarily equate to better, emphasizing that human visual experiences may not directly translate to machine vision tasks. Additionally, the inference time of the SOTA is contingent on the downstream task (i.e., 11.682s with lightweight YOLO-v3). In contrast, our approach's inference time remains unaffected by the downstream task, consistently achieving a swift 3.255ms. The high efficiency demonstrated, independent of downstream tasks, further underscores the strengths of our design.

Table 3: The mean average precision of object detection.

Work	Person↑	Car↑	Bus↑	mAP↑	Time↓
Default	0.3630	0.6247	0.3255	0.4377	-
Experts	0.2995	0.5791	0.2260	0.3682	4h
TOG'19 [15]	0.3338	0.5758	0.2058	0.3718	11.682s
Ours (U-Net proxy)	0.3003	0.5921	0.2834	0.3919	**3.255ms**
Ours (Our HFP)	**0.3972**	**0.7092**	**0.3938**	**0.5001**	

Table 4: The objective and perceptual quality of robustness.

Work	LPIPS↓	PSNR↑	SSIM↑	NIQE↓
Default	0.7253	14.1785	0.3071	12.4161
Experts	0.4102	20.2171	0.8005	6.7606
TOG'19 [15]	0.4870	18.5020	0.7131	7.7295
Ours	**0.3851**	**22.7684**	**0.8130**	**6.2629**

Robustness. To evaluate the robustness of our framework against real-world changes, we conduct tests on unforeseen scenarios. Given that human experts and the SOTA do not support online search, we simplify the difficulty by selecting a subset of 10 parameters to ensure fairness in our evaluation. As depicted in Figure 6, images processed by ISP using our approach maintain clarity in nighttime scenes, while the SOTA and human experts struggle to adapt to the transition from daytime to nighttime, resulting in less clear images. In the comparison of objective and perceptual quality in unforeseen scenes, it is evident that our method consistently delivers superior quality compared to the SOTA and experts.

5.3 Implementation and Complexity

Here we present our hardware implementation and evaluate its complexity.

Implementation. We implement the entire real-time hardware design using Verilog and HLS for ParaNet and *xkISP*-Lite, respectively, on both FPGA (xc7z035fbg676) and ASIC (GF 28nm). Figure 7 illustrates our system implementation on an FPGA. We adopt Sony imx485 camera sensor. The ParaNet is with 10-bit quantization, and the ISP uses a 12-bit data width. Under 10-bit quantization, our approach has only a loss of 0.18dB in PSNR in human vision and a loss of 0.04 in mAP in object detection, yet it still maintains

Table 5: Hardware Implementation under 1080P

Tech	Module	LUTs/FFs/BRAMs/DSPs(Area)	Clock(Hz)	FR
FPGA	ParaNet	37692 / 78936 / 36.5 / 365	156.25M	75FPS
	xkISP-Lite	17212 / 14832 / 25.5 / 127		
	Total	**54904 / 93768 / 62 / 492**		
ASIC	ParaNet	$0.7446mm^2$	500M	240FPS
	xkISP-Lite	$0.5514mm^2$		
	Total	$\mathbf{1.2960}mm^2$		

Figure 6: The visual comparison for robustness test.

Figure 7: The implementation of Auto-ISP on FPGA.

Table 6: The complexity of the neural network.

Network	Our ParaNet	U-Net Proxy	Our Proxy
Parameter(M)	0.0093	4.3707	2.8399
Operations (GFLOPs)	0.0157	21.1975	98.8356

Table 7: The ablation study of ParaNet designing in FPGA.

Semi-Parallel	Row-Cross	PS Conv	LUTs/FFs/BRAMs/DSPs↓
×2	✓	✓	46935 / 90834 / 36.5 / 721
×1		✓	38384 / 79570 / 32.5 / 383
×1	✓		38877 / 80588 / 37 / 413
×1	✓	✓	37692 / 78936 / 36.5 / 365

optimal performance. We present the results of our implementation in Table 5. In our FPGA and ASIC implementation, we achieve frame-by-frame search and imaging at 1080P at 75FPS and 240FPS, respectively.

Complexity. As indicated in Table 6, our hardware-friendly ParaNet design use only 0.0093M parameters, effectively reducing the memory requirement. Furthermore, our proxy features fewer parameters (~2.84M) and enhanced floating-point computational capabilities (~98.84GFLOPs) compared to a U-Net-based proxy.

Table 7 further validate the reduction of ParaNet hardware resources for our approach. The semi-parallel architecture serially computes the ×1 output channel/node. When computing with ×2 outputs, ParaNet has more Conv kernels, with a significant increase in DSPs. Row-Cross Conv and PS Conv also reduce logic resources, especially for DSPs.

6 Conclusion

We introduce a tri-step framework for solving real-time online ISP hyperparameter optimization tasks, greatly simplifying the complex tuning process and eliminating the need for manual intervention. We integrate the framework and ISP hardware into Auto-ISP and propose a semi-parallel architecture and row-aware processing to improve compatibility with conventional ISPs. We validate our design on FPGAs and ASICs, and results show that our approach achieves the highest accuracy in both HV and CV tasks and can be applied in real-time even in unpredictable scenarios.

References

[1] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. 2019. Unprocessing images for learned raw denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11036–11045.

[2] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. 2018. Learning to see in the dark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3291–3300.

[3] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pająk, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. 2014. Flexisp: A flexible camera image processing framework. *ACM Transactions on Graphics (ToG)* 33, 6 (2014), 1–13.

[4] Yingkun Hou, Jun Xu, Mingxia Liu, Guanghai Liu, Li Liu, Fan Zhu, and Ling Shao. 2020. NLH: A blind pixel-level non-local method for real-world image denoising. *IEEE Transactions on Image Processing* 29 (2020), 5121–5135.

[5] Nasir Memon, David L Neuhoff, and Sunil Shende. 2000. An analysis of some common scanning techniques for lossless image coding. *IEEE transactions on image processing* 9, 11 (2000), 1837–1848.

[6] Ali Mosleh, Avinash Sharma, Emmanuel Onzon, Fahim Mannan, Nicolas Robidoux, and Felix Heide. 2020. Hardware-in-the-loop end-to-end optimization of camera image processing pipelines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7529–7538.

[7] OpenASIC. 2022. xkISP. https://github.com/openasic-org/xkISP.

[8] Guocheng Qian, Yuanhao Wang, Jinjin Gu, Chao Dong, Wolfgang Heidrich, Bernard Ghanem, and Jimmy S Ren. 2022. Rethinking learning-based demosaicing, denoising, and super-resolution pipeline. In *2022 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 1–12.

[9] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[10] Nicolas Robidoux, Luis E Garcia Capel, Dong-eun Seo, Avinash Sharma, Federico Ariza, and Felix Heide. 2021. End-to-end high dynamic range camera pipeline optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6297–6307.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241.

[12] Denis Schapiro, Artem Sokolov, Clarence Yapp, Yu-An Chen, Jeremy L Muhlich, Joshua Hess, Allison L Creason, Ajit J Nirmal, Gregory J Baker, Maulik K Nariya, et al. 2022. MCMICRO: a scalable, modular image-processing pipeline for multiplexed tissue imaging. *Nature methods* 19, 3 (2022), 311–315.

[13] Yongjie Shi, Songjiang Li, Xu Jia, and Jianzhuang Liu. 2022. Refactoring ISP for High-Level Vision Tasks. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2366–2372.

[14] Ethan Tseng, Ali Mosleh, Fahim Mannan, Karl St-Arnaud, Avinash Sharma, Yifan Peng, Alexander Braun, Derek Nowrouzezahrai, Jean-Francois Lalonde, and Felix Heide. 2021. Differentiable compound optics and processing pipeline optimization for end-to-end camera design. *ACM Transactions on Graphics (TOG)* 40, 2 (2021), 1–19.

[15] 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 proxies. *ACM Trans. Graph.* 38, 4 (2019), 27–1.

[16] Xiaodong Yang, Wengang Zhou, and Houqiang Li. 2020. MCFD: A hardware-efficient noniterative multicue fusion demosaicing algorithm. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 9 (2020), 3575–3589.

[17] Ke Yu, Zexian Li, Yue Peng, Chen Change Loy, and Jinwei Gu. 2021. Reconfigisp: Reconfigurable camera image processing pipeline. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4248–4257.

[18] Xiaoming Zeng, Zhendong Wang, and Yang Hu. 2022. Enabling efficient deep convolutional neural network-based sensor fusion for autonomous driving. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 283–288.

[19] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. 2021. Designing a practical degradation model for deep blind image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4791–4800.