POS-ISP: Pipeline Optimization at the Sequence Level for Task-aware ISP

— Supplementary Material —

Contents

S1. Additional Results

S1.1. Additional Qualitative Results

Additional qualitative examples for object detection and instance segmentation are shown in Fig. S1, and additional results for image enhancement are provided in Fig. S5. These results provide further visual references that complement the comparisons presented in the main paper.

S1.2. Image Enhancement

We provide quantitative comparisons for the image enhancement task described in the main paper. The results are summarized in Tab. S1. Existing task-driven ISP optimization methods improve quantitative metrics compared to the input images, indicating that optimizing the ISP pipeline with task supervision can benefit perceptual enhancement. However, their performance remains limited, and the outputs often fail to fully reproduce the desired style. DRL-ISP [12] tends to uniformly increase image intensity due to unstable policy learning, leaving some regions under-enhanced. ReconfigISP [15] applies a fixed pipeline to all images, which often produces overly strong contrast and tone deviations from the target style. AdaptiveISP [14] achieves relatively strong performance but fails to consistently find the optimal configurations, resulting in color and white balance shifts. In contrast, POS-ISP achieves the best performance across the reported metrics. This result suggests that our sequence-level optimization framework can effectively learn ISP configurations that align with the target retouching style.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Input RAW	9.18	0.195	0.585
DRL-ISP [12]	13.68	0.571	0.492
ReconfigISP [15]	17.28	0.818	0.216
AdaptiveISP [14]	22.73	0.908	0.105
Ours	**23.11**	**0.923**	**0.097**

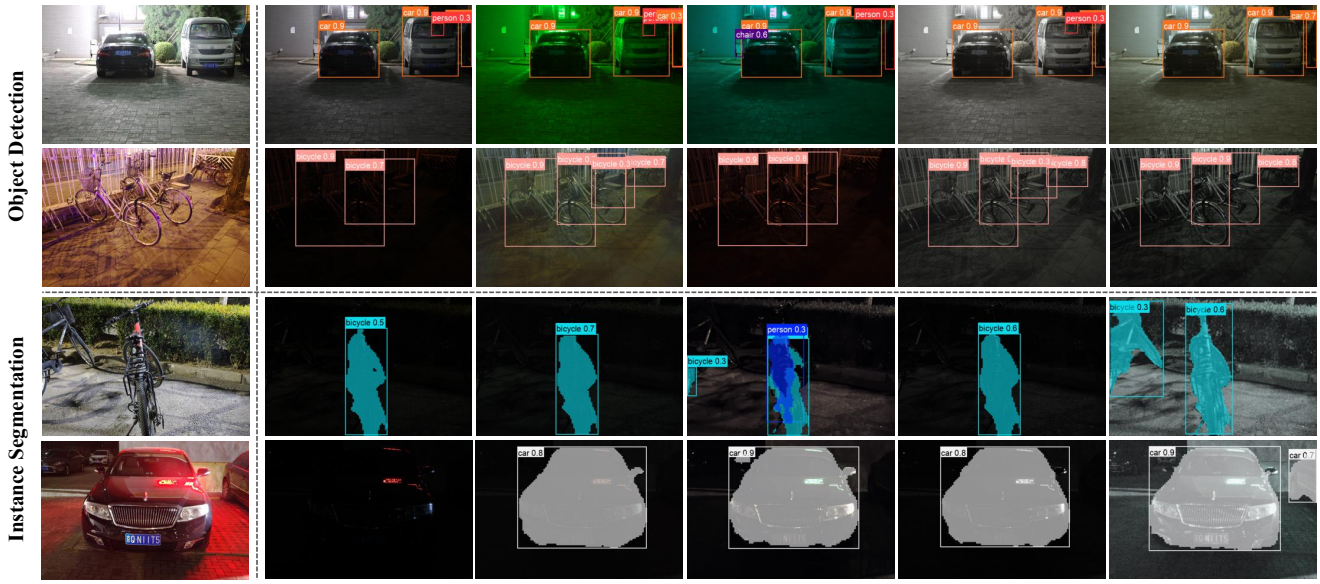
Table S1. **Quantitative comparison for full-reference image quality enhancement task.** In the table, "Input RAW" denotes images that have undergone only the most basic preprocessing steps using Adobe Lightroom. We highlight the best metrics as bold.

Method	**Error ↓**				**Accuracy ↑**
	AbsRel	SqRel	RMS	RMSlog	$\delta < 1.25$
Input RAW	0.293	2.560	8.73	0.398	0.493
DRL-ISP [12]	0.131	0.978	5.22	0.209	0.839
ReconfigISP [15]	0.154	1.128	5.76	0.235	0.788
AdaptiveISP [14]	0.131	0.971	5.23	0.207	0.840
Ours	**0.128**	**0.947**	**5.13**	**0.205**	**0.847**

Table S2. **Quantitative comparison with depth estimation task.** We highlight the best metrics as bold.

S1.3. Depth Estimation

We further evaluate POS-ISP on monocular depth estimation to examine its effectiveness in dense geometric prediction tasks. In this setting, the task loss $\mathcal{L}_{\mathcal{T}}$ is defined using the Root Mean Squared Error (RMSE) objective, which emphasizes large depth discrepancies and penalizes outliers. Following DRL-ISP [12], depth supervision is obtained from SC-SfMLearner [1] predictions on the



(a) Reference (b) Camera ISP (c) DRL-ISP (d) ReconfigISP (e) AdaptiveISP (f) POS-ISP (Ours)

Figure S1. **Comparison of different ISP methods on object detection and instance segmentation tasks.** Reference images are from the LOD-Normal and LIS-Normal datasets with brightness increased by $1.5\times$ for visualization.

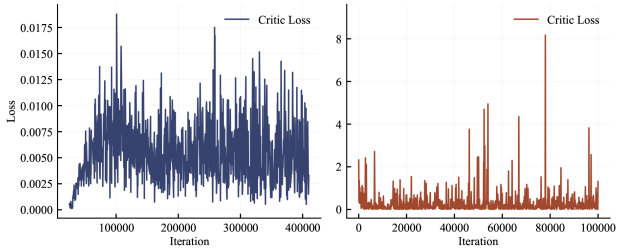
KITTI [4] dataset. For fair comparison, the same task loss and depth predictor are consistently applied to all methods. Performance is evaluated using standard depth estimation metrics, including AbsRel, SqRel, RMS, RMSlog, and accuracy ($\delta < 1.25$).

Tab. S2 presents the results when models are trained using the RMSE objective. Compared to existing ISP optimization methods, POS-ISP consistently achieves the best performance across all evaluation metrics. These improvements are observed not only in RMS but also in relative error metrics and accuracy, indicating that the gains are not restricted to a single evaluation criterion.

Overall, these results demonstrate that the performance gains of POS-ISP are not confined to recognition-based tasks. Instead, the method generalizes effectively to dense geometric prediction, highlighting its robustness and effectiveness across diverse downstream tasks, including detection, segmentation, image enhancement, and depth estimation.

Method	Dataset	Object Detection (LOD)			Instance Segmentation (LIS)		
		mAP @0.5:0.95	mAP @0.5	mAP @0.75	mAP @0.5:0.95	mAP @0.5	mAP @0.75
AdaptiveISP	Dark	47.2	71.4	51.7	25.2	42.3	25.2
Ours	Dark	**47.8** $\pm$ **0.08**	**72.1** $\pm$ **0.16**	**52.7** $\pm$ **0.12**	**32.2** $\pm$ **0.05**	**51.9** $\pm$ **0.09**	**32.2** $\pm$ **0.05**
AdaptiveISP	All	56.1	72.8	60.6	32.4	52.3	32.5
Ours	All	**56.4** $\pm$ **0.20**	**73.1** $\pm$ **0.15**	**60.7** $\pm$ **0.21**	**34.9** $\pm$ **0.45**	**55.6** $\pm$ **0.66**	**34.8** $\pm$ **0.45**

Table S3. **Multi-seed results compared with the strongest baseline (AdaptiveISP [14]).** POS-ISP consistently outperforms AdaptiveISP across both data conditions and tasks, while maintaining low variance across seeds.



(a) DRL-ISP (b) AdaptiveISP

Figure S2. **Training loss trajectories of critic networks in AdaptiveISP [14] and DRL-ISP [12].** The graphs show unstable oscillations and spikes during actor–critic training.

S2. Additional Analysis

S2.1. Optimization Stability

Multi-seed experiments To further analyze training stability, we report results over three random seeds for both object detection and instance segmentation tasks and compare the results with AdaptiveISP [14]. As shown in Tab. S3, POS-ISP consistently outperforms the baseline across different random seeds, indicating stable optimization and reproducible gains.

Instability of the critic network POS-ISP adopts sequence-level optimization to mitigate the unstable optimization behavior observed in prior stepwise RL-based ISP methods [12, 14]. In these methods, the instability primarily arises from the critic network used to estimate future rewards. To demonstrate the instability of the critic network adopted by other RL-based methods, we visualize the loss graphs of the critic networks of DRL-ISP [12] and AdaptiveISP [14] in Fig. S2. As shown in the figure, the learn-

ing curves exhibit oscillations and spikes during training, as errors in the bootstrapped critic estimates accumulate and propagate through the policy updates, leading to unstable optimization.

Such oscillatory behavior has also been reported in prior RL works, where bootstrapped value errors and recurrent policy oscillations are identified as key sources of instability [7, 9, 13]. In contrast, POS-ISP reformulates the problem as sequence-level optimization, eliminating the need for an unstable critic network.

S2.2. Runtime Analysis

End-to-end runtime We analyze the total inference time, including both pipeline estimation and execution, for models trained on the LOD-Dark [5] dataset with a YOLOv3 [11] detector. We compare POS-ISP with DRL-ISP [12] and AdaptiveISP [14]. The results are reported in Tab. S4. POS-ISP achieves the best detection performance among the compared methods. However, because POS-ISP predicts ISP pipelines without strict structural constraints, the resulting pipelines tend to be longer, leading to increased inference latency. Consequently, its end-to-end runtime is higher than that of prior methods, which explicitly constrain pipeline length.

To reduce this overhead, we can introduce an additional runtime penalty during training, following the strategy used in AdaptiveISP. With this penalty, the predicted pipelines become significantly shorter, leading to a substantial reduction in end-to-end latency. Importantly, this reduction in runtime is achieved while maintaining detection performance comparable to that of the default POS-ISP configuration. These results demonstrate that POS-ISP provides a flexible mechanism for balancing accuracy and efficiency. By incorporating a runtime-aware training objective, the framework can produce shorter pipelines and improve runtime efficiency without significant degradation in task performance.

On-device runtime analysis We further evaluate the runtime on a Galaxy S10 CPU to assess practical deployability. All measurements are conducted in FP32 precision using a single CPU thread, without quantization or hardware-specific acceleration. As shown in Tab. S5, POS-ISP runs about $4\times$ faster than AdaptiveISP on device. These results demonstrate that POS-ISP maintains high computational efficiency even on resource-constrained mobile hardware, enabling real-time operation on an older mobile CPU.

S2.3. Fixed vs. Dynamic Pipeline Length

To demonstrate that the performance gains of POS-ISP stem from its optimization strategy rather than increased pipeline length, we compare it with a fixed-length variant and further examine whether dynamic length adaptation provides additional improvements. Following AdaptiveISP [14], which models the pipeline as five processing stages, we train POS-ISP with the same fixed sequence length and candidate module set. The results are summarized in Tab. S6. Even under the fixed-length setting, POS-ISP outperforms AdaptiveISP, indicating that its performance gain arises from sequence-level optimization and stable training guided by final-output rewards, rather than from simply employing a longer pipeline. Allowing the pipeline length to be determined dynamically yields further improvements, consistently surpassing both AdaptiveISP and the fixed-length variant.

Method	LOD-Dark (YOLOv3)			Runtime (ms)		
	mAP @0.50:0.95	mAP @0.5	mAP @0.75	Prediction	End-to-end	Length
DRL-ISP [12]	44.2	67.8	48.4	15.71	26.85	3
AdaptiveISP [14]	47.1	71.4	51.7	12.72	<u>16.30</u>	5
Ours	**47.8**	**72.1**	**52.8**	**1.55**	82.57	10
Ours (w/ penalty)	<u>47.7</u>	<u>72.0</u>	<u>52.2</u>	**1.55**	**1.61**	3

Table S4. **Accuracy–runtime trade-off with runtime penalty on LOD-Dark.** Object detection performance and corresponding runtime measured on a single NVIDIA RTX 2080 Ti with 512×512 input images.

Method	Prediction (ms / FPS)	End-to-end (ms / FPS)
AdaptiveISP [14]	29.0 / 34.48	115.0 / 8.70
Ours (w/ penalty)	**7.21 / 138.70**	**32.4 / 30.86**

Table S5. **On-device runtime comparison on a Galaxy S10 CPU.** Runtime is measured using ISP pipelines trained on the LOD-Dark dataset with 512×512 input images.

Method	Length Constraint	LOD-Dark			LIS-Dark		
		mAP @0.5:0.95	mAP @0.5	mAP @0.75	mAP @0.5:0.95	mAP @0.5	mAP @0.75
AdaptiveISP [14]	5	47.1	71.4	<u>51.7</u>	25.2	42.3	25.2
Ours	5	<u>47.4</u>	<u>71.7</u>	51.6	<u>31.0</u>	<u>50.7</u>	<u>31.1</u>
	$\leq$10	**47.8**	**72.1**	**52.8**	**32.1**	**51.8**	**32.1**

Table S6. **Ablation study on the impact of fixed and dynamic pipeline length.** We highlight the best metrics as bold and the second best metrics as underline.

S2.4. Single Sequence per Task

POS-ISP employs a single sequence per task while adapting module parameters on a per-image basis. In this framework, the sequence defines the overall processing structure of the pipeline, while the parameters control image-specific behavior within that structure. Although the sequence remains fixed, adapting parameters per image provides sufficient flexibility to accommodate diverse inputs.

Experimental observations support this behavior. As shown in Tab. S7, our method achieves the lowest degradation rate $P(\Delta < 0)$ and the mildest worst-case drops across both datasets, demonstrating stable behavior across diverse inputs despite using a single sequence.

To further analyze robustness across different illumi-

Method	LOD-All				LIS-All			
	$P(\Delta<0)\downarrow$			Worst 5% Mean↑	$P(\Delta<0)\downarrow$			Worst 5% Mean↑
	All	Normal	Dark		All	Normal	Dark	
DRL-ISP [12]	0.185	0.186	0.184	-0.28	0.423	0.441	0.405	-0.63
ReconfigISP [15]	0.252	0.229	0.274	-0.31	0.287	0.309	0.265	-0.38
AdaptiveISP [14]	0.135	**0.138**	0.132	-0.18	0.269	0.217	0.321	-0.30
Ours	**0.123**	**0.138**	**0.108**	**-0.16**	**0.187**	**0.199**	**0.174**	**-0.25**

Table S7. **Output-level robustness analysis using per-image ΔAP@0.5 (w.r.t. input RAW).** $P(\Delta < 0)$ denotes the fraction of images whose performance degrades after ISP, and Worst 5% Mean reports the average drop among the most severely degraded cases (AP in the [0,1] scale). Best results are in bold.

Method	LOD-Dark			LOD-All		
	mAP @0.5:0.95	mAP @0.5	mAP @0.75	mAP @0.5:0.95	mAP @0.5	mAP @0.75
Input RAW	34.8	53.8	37.5	44.3	62.2	46.5
Camera ISP	26.9	41.4	28.9	40.1	56.2	42.4
DRL-ISP [12]	35.4	54.7	38.0	44.7	62.5	46.7
ReconfigISP [15]	33.4	52.0	35.9	14.0	20.7	14.6
AdaptiveISP [14]	36.8	56.1	40.2	47.3	65.4	49.9
Ours	**38.1**	**57.9**	**41.7**	**48.1**	**65.5**	**51.0**

Table S8. **Quantitative comparison with object detection task with YOLOv13.** In the table, "Input RAW" denotes images that have undergone only the most basic preprocessing steps: defective pixel correction, black level correction, lens shading correction, and demosaicking. We highlight the best metrics as bold.

nation conditions, we additionally report $P(\Delta < 0)$ on the Normal (well-lit) and Dark (low-light) subsets in the LOD [5] and LIS [3] datasets. Our method shows comparable degradation rates across the two subsets (LOD: 0.138 vs. 0.108; LIS: 0.199 vs. 0.174), indicating that the optimized pipeline behaves consistently across lighting variations.

These results indicate that a fixed pipeline sequence combined with per-image parameter adaptation provides sufficient flexibility to handle diverse image conditions while maintaining stable behavior across the dataset. If a single shared sequence were fundamentally insufficient, we would expect significantly higher degradation rates or substantially worse worst-case drops across subsets, which is not observed in our results.

S2.5. Generalization Across Optimization Settings

To evaluate the robustness of POS-ISP beyond a specific optimization setting, we analyze its behavior across different reward backbones and training objectives.

Reward backbone We evaluate generalization across different detection backbones used for reward computation. While the main paper computes task rewards using a single pretrained detector, we conduct additional experiments with YOLOv13 [8]. As reported in Tab. S8, ReconfigISP [15] performs worse than the input images due to the mismatch between soft training and hard inference. On LOD-All, the optimized pipeline tends to overfit the Dark subset, resulting in overly aggressive enhancement that causes Normal images to become saturated. Because this single fixed pipeline is applied uniformly to all inputs without adapting to individual image characteristics, the performance degradation becomes significant. Other RL-based methods [12, 14] improve upon the input images but remain suboptimal, as their stepwise optimization depends on unstable estimation of future rewards. In contrast, POS-ISP achieves the best performance with YOLOv13, demonstrating strong generalization across different pretrained reward models. Overall, these results show that sequence-level joint optimization provides a stable and backbone-agnostic framework that remains effective under varying training objectives and reward models.

Loss	Method	Error ↓			Accuracy ↑
		AbsDiff	AbsRel	RMS	$\delta < 1.25$
RMSE	AdaptiveISP	2.574	0.131	5.22	0.840
	Ours	**2.525**	**0.128**	**5.13**	**0.847**
AbsDiff	AdaptiveISP	2.548	0.128	6.24	0.847
	Ours	**2.504**	**0.127**	**5.15**	**0.850**
AbsRel	AdaptiveISP	2.656	0.134	5.38	0.833
	Ours	**2.522**	**0.126**	**5.19**	**0.847**

Table S9. **Comparison under different training objectives.** For each objective setting, POS-ISP consistently outperforms AdaptiveISP [14] across all metrics.

Training objective To evaluate robustness to different training objectives, we train POS-ISP and AdaptiveISP [14] on the depth estimation task using different loss functions, including Root Mean Squared Error (RMSE), Absolute Difference (AbsDiff), and Absolute Relative Error (AbsRel), under the same SC-SfMLearner [1] supervision. These objectives introduce different optimization biases: RMSE emphasizes large depth errors, AbsDiff measures average absolute deviation, and AbsRel captures scale-aware discrepancies.

As shown in Tab. S9, POS-ISP consistently outperforms AdaptiveISP across all training objectives. The performance gains remain stable across different loss functions and evaluation metrics, indicating that our method does not depend on a specific objective and exhibits robust optimization behavior under diverse depth supervision signals.

S2.6. Ablation on Parameter Predictor

We observe that conditioning the parameter predictor solely on the input image is sufficient in our training setting. Early in training, the sequence policy exhibits high entropy and explores a wide range of ISP pipelines. In this regime, an image-only predictor (IO) learns parameters over transient and rapidly changing sequences. As training progresses, the policy entropy decreases and the sequence distribution gradually concentrates on a small set of high-reward pipelines. Consequently, the predictor ultimately needs to

Parameter Predictor	LOD-Dark			LIS-Dark		
	mAP @0.5:0.95	mAP @0.5	mAP @0.75	mAP @0.5:0.95	mAP @0.5	mAP @0.75
(a) SC	47.5 ± 0.26	71.5 ± 0.28	52.0 ± 0.42	31.6 ± 0.33	51.2 ± 0.42	31.3 ± 0.52
(b) IO (Ours)	$\mathbf{47.8 \pm 0.08}$	$\mathbf{72.1 \pm 0.16}$	$\mathbf{52.7 \pm 0.12}$	$\mathbf{32.2 \pm 0.05}$	$\mathbf{51.9 \pm 0.09}$	$\mathbf{32.2 \pm 0.05}$

Table S10. **Comparison between sequence-conditioned (SC) and image-only (IO) parameter predictors.** We report mean $\pm$ standard deviation over three seeds.

Source Task (Sequence)	mAP@0.5:0.95	mAP@0.5	mAP@0.75
Image Enhancement	47.6	71.7	52.4
Instance Segmentation	47.1	71.4	50.9
Object Detection	**48.0**	**72.2**	**52.8**

Table S11. **Cross-task evaluation.** Module sequences from different source tasks, evaluated on object detection with parameters retrained for that task. We highlight the best metrics as bold.

model only the final converged pipeline, making image-only conditioning adequate in practice.

For comparison, we implement a sequence-conditioned variant (SC), where sequence features are modulated by image features and the latent dimension is increased by a factor of two, while keeping the rest of the architecture unchanged. Although SC is more expressive in principle, rapidly changing sequences during early training introduce unstable conditioning that interferes with parameter learning. As shown in Tab. S10, SC exhibits higher variance and lower performance across random seeds compared to IO. These results suggest that additional conditioning complexity in the predictor does not necessarily improve optimization in our sequence-level training framework.

S2.7. Importance of Task-adaptive Sequence

We train the sequence predictor to adapt its sequence to the target task. To assess the importance of such task-aware sequences in downstream task performance, we conduct an experiment. In this experiment, we first obtain three sequences that are optimized for image enhancement, instance segmentation, and object detection, respectively. Then, we construct three ISP pipelines using the sequences and optimize only the parameter predictor on the object detection task using the LOD-Dark [5] dataset and YOLOv3 [11] detector. By optimizing only the module parameters while keeping the module sequences fixed, we can examine the effect of the module sequence on the overall ISP pipeline.

The results are reported in Tab. S11. As shown in the table, the sequence optimized for other tasks yields noticeably lower performance when applied to the detection task, even after its parameters are retrained. In contrast, the sequence originally optimized for the same task, object detection, preserves strong performance. These results indicate that the module sequence itself plays a decisive role in downstream performance and cannot be recovered through parameter optimization alone. This finding underscores the necessity of task-aware sequence design, as different tasks naturally favor different ISP module orders.

Estimated from	Applied to	mAP@0.5:0.95	mAP@0.5	mAP@0.75
LOD-Normal	LOD-Dark	52.6	70.0	57.0
LOD-Dark	LOD-Dark	**53.6**	**71.2**	**57.4**

Table S12. **Cross-exposure evaluation of ISP parameter transfer between different datasets.** We highlight the best metrics as bold.

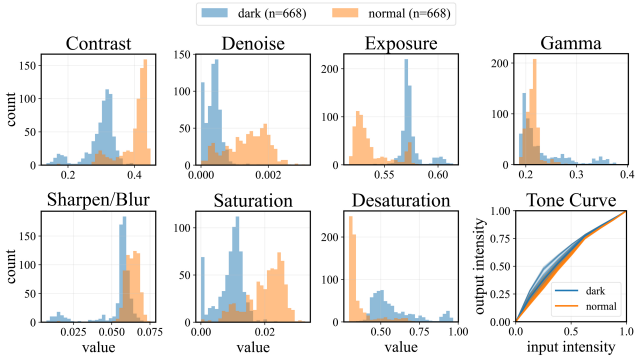
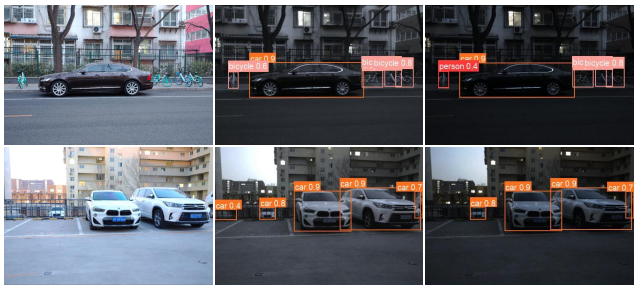


Figure S3. **Histogram comparisons of the predicted parameters across illumination domains.**



(a) Reference (b) w/ Parameters from Dark Image (c) w/ Parameters from Normal Image

Figure S4. **Qualitative examples of cross-domain parameter swapping.** Reference images are well-lit scenes from the LOD-Normal dataset, with brightness increased by $1.5\times$ for visualization. ISP parameters are estimated from different domains and applied to LOD-Dark images.

S2.8. Image-adaptive Parameter Prediction

Our parameter predictor conditions on the input image to produce image-adaptive module parameters. To examine whether these parameters truly respond to scene characteristics, particularly illumination, we design a cross-domain parameter-swap experiment. We use our POS-ISP model trained on LOD-All [5] with the YOLOv3-based [11] detection loss. Using this model, we estimate ISP parameters from images in either LOD-Dark or LOD-Normal. We then apply these parameters to LOD-Dark images and evaluate the resulting performance.

Tab. S12 shows the quantitative results. When parameters are predicted and applied within the same image

domain, the model achieves the highest performance in each case. However, when parameters are transferred across domains, performance consistently drops. This degradation confirms that the predicted parameters are sensitive to scene conditions and that the parameter predictor effectively adapts its predictions to the input image.

We additionally visualize the distributions of the predicted parameters in Fig. S3 as well as the output images in Fig. S4. When parameters estimated from LOD-Normal images are applied to LOD-Dark scenes, the resulting outputs become underexposed or locally flattened, clearly reflecting the domain mismatch. The parameter histograms also reveal noticeable shifts between Dark and Normal inputs: exposure- and tone-related parameters predicted for LOD-Dark images tend to apply stronger brightness compensation, whereas those predicted for LOD-Normal images show more moderate adjustments. This separation in parameter distributions indicates that the parameter predictor does not regress to a domain-agnostic average but instead adapts its predictions to the illumination characteristics of each input. Together with the performance drop observed under cross-domain transfer, these qualitative and statistical findings further demonstrate that POS-ISP learns image-aware parameter prediction closely aligned with scene conditions.

S3. Implementation Details

S3.1. ISP Modules

The parameter predictor predicts parameter vectors for the ISP modules, where the dimensionalities are module-specific and the elements are normalized to the range of $[0, 1]$. For a module $\mathcal{M}_i$, we denote its parameters by $\theta_i \in [0, 1]^{d_i}$, where d_i is the number of parameters for that module. These normalized parameters are then rescaled to their corresponding module-specific numerical ranges and used to parameterize the modules. Following AdaptiveISP [14], we have 10 candidate modules from which modules are selected to construct an ISP pipeline. In the following, we describe the modules that constitute our module pool, their formulations, and the parameter ranges of each module.

In the following section, we denote an RGB pixel $\mathbf{I}$ as

$$\mathbf{I} = (I_R,\, I_G,\, I_B)^\top, \tag{1}$$

where I_R, I_G, and I_B represent the intensities of the red, green, and blue channels, respectively. We also denote an output RGB pixel of a module as $\mathbf{I}'$.

Exposure control For a module for exposure control, the parameter predictor predicts $\theta_e \in [0, 1]$. The parameter θ_e is then rescaled to $\theta'_e \in [-3.5, 3.5]$ and used to adjust the brightness of the input pixel $\mathbf{I}$:

$$\mathbf{I}' = 2^{\theta'_e}\,\mathbf{I}. \tag{2}$$

Gamma correction The parameter predictor predicts $\theta_g \in [0, 1]$ for the gamma correction module. The parameter θ_g is then rescaled to $\theta'_g \in \left[\frac{1}{3}, 3\right]$, which controls the gamma curve for nonlinear brightness shaping:

$$\mathbf{I}' = \mathbf{I}^{\theta'_g}. \tag{3}$$

Tone mapping We first define eight piecewise-linear basis functions that construct a tone-mapping curve:

$$b_i(u) = \max\left(0,\ \min\left(u - \frac{i-1}{8},\ \frac{1}{8}\right)\right), \tag{4}$$

where $i = 1, ..., 8$. For a module for tone mapping, the parameter predictor predicts $\theta_t \in [0, 1]^8$, which is rescaled to $\theta'_t \in [0.5, 2.0]^8$. Then, the tone mapping curve is applied as:

$$\mathbf{I}' = \frac{8}{Z}\sum_{i=1}^{8} \theta'_{t,i}\, b_i(\mathbf{I}), \qquad Z = \sum_{i=1}^{8} \theta'_{t,i}, \tag{5}$$

where $\theta'_{t,i}$ is i-th element of θ'_t.

Contrast The parameter predictor predicts a parameter $\theta_c \in [0, 1]$, which is rescaled to $\theta'_c \in [-1, 1]$ and used as a contrast-adjustment factor. The module enhances contrast by blending the original RGB vector with an S-curve–reshaped version of the image.

The luminance is computed as

$$I_{\text{lum}} = 0.27 I_R + 0.67 I_G + 0.06 I_B, \tag{6}$$

and its S-curve transformation and RGB scaling are defined as

$$\tilde{\mathbf{I}} = \mathbf{I} \cdot \frac{I_{\text{S}}}{I_{\text{lum}} + \varepsilon}, \qquad I_{\text{S}} = \frac{1 - \cos(\pi I_{\text{lum}})}{2}, \tag{7}$$

where ε prevents division by zero. Finally, the contrast-adjusted output is

$$\mathbf{I}_{\text{contrast}} = (1 - \theta'_c)\,\mathbf{I} + \theta'_c\,\tilde{\mathbf{I}}. \tag{8}$$

Saturation To control saturation, we first convert $\mathbf{I}$ into HSV color space, yielding

$$\mathbf{I}_{\text{HSV}} = (H, S, V)^\top. \tag{9}$$

Then, we strengthen the saturation via

$$S_{\text{enh}} = S + 0.8 \cdot (1 - S)(0.5 - |0.5 - V|)\,. \tag{10}$$

The saturated pixel is converted to RGB color space again, yielding

$$\mathbf{I}_{\text{enh}} = (I_R^{enh}, I_G^{enh}, I_B^{enh})^\top. \tag{11}$$

The output pixel is then blended with the saturated pixel as following:

$$\mathbf{I}' = (1 - \theta_s)\,\mathbf{I} + \theta_s\,\mathbf{I}_{\text{enh}}, \tag{12}$$

where $\theta_s \in [0, 1]$ is predicted by the parameter predictor.

Module	Equation	Parameter Type	# Params	Parameter Range
Exposure control	$\mathbf{I}' = 2^{\theta'_e}\,\mathbf{I}$	EV shift θ'_e	1	$\theta'_e \in [-3.5, 3.5]$
Gamma correction	$\mathbf{I}' = \mathbf{I}^{\theta'_g}$	Gamma exponent θ'_g	1	$\theta'_g \in [\frac{1}{3}, 3]$
Tone mapping	$\mathbf{I}' = \frac{8}{Z}\sum_{i=1}^{8} \theta'_{t,i}\, b_i(\mathbf{I})$	Segment weights $\theta'_{t,i}$	8	$\theta'_{t,i} \in [0.5, 2.0]$
Contrast	$\mathbf{I}' = (1-\theta'_c)\,\mathbf{I} + \theta'_c\,\tilde{\mathbf{I}}$	Contrast factor θ'_c	1	$\theta'_c \in [-1, 1]$
Saturation	$\mathbf{I}' = (1-\theta_s)\,\mathbf{I} + \theta_s\,\mathbf{I}_{\text{enh}}$	Blend factor θ_s	1	$\theta_s \in [0, 1]$
Desaturation	$\mathbf{I}' = (1-\theta_d)\,\mathbf{I} + \theta_d\,\mathbf{I}_{\text{gray}}$	Blend factor θ_d	1	$\theta_d \in [0, 1]$
White balance	$\mathbf{I}' = \text{diag}(\tilde{\theta}'_{wb,R}, \tilde{\theta}'_{wb,G}, \tilde{\theta}'_{wb,B})\,\mathbf{I}$	Normalized gains $\tilde{\theta}'_{wb,c}$	3	$\theta'_{wb,c} \in [1/1.1,\ 1.1]$
Denoise	$\mathbf{I}' = \frac{\sum_{y\in\mathcal{N}(x)} w(x,y\mid\theta'_d)\,\mathbf{I}(y)}{\sum_{y\in\mathcal{N}(x)} w(x,y\mid\theta'_d)}$	Denoise strength θ'_d	1	$\theta'_d \in [10^{-5}, 1]$
Sharpen/Blur	$\mathbf{I}' = \theta'_{sh}\,\mathbf{I} + (1-\theta'_{sh})\,\mathbf{I}_{\text{blur}}$	Sharpen/blur factor θ'_{sh}	1	$\theta'_{sh} \in [10^{-5}, 10]$
Color Correction	$\mathbf{I}' = M\,\mathbf{I}$	Color correction matrix entries θ'_{ccm}	9	$\theta'_{ccm,ij} \in [-2, 2]$

Table S13. **Details of the adopted ISP modules.**

Desaturation This module reduces colorfulness by blending the RGB vector with its grayscale version. The parameter predictor predicts a parameter $\theta_d \in [0,1]$, which controls how strongly the pixel is pulled toward a monochrome appearance.

The luminance and the corresponding grayscale RGB vector are defined as

$$\begin{aligned} I_{\text{lum}} &= 0.27 I_R + 0.67 I_G + 0.06 I_B, \\ \mathbf{I}_{\text{gray}} &= (I_{\text{lum}},\, I_{\text{lum}},\, I_{\text{lum}})^\top, \end{aligned} \tag{13}$$

and the output is produced by linearly interpolating between the original color and its grayscale representation:

$$\mathbf{I}' = (1-\theta_d)\,\mathbf{I} + \theta_d\,\mathbf{I}_{\text{gray}}. \tag{14}$$

White balance For white balancing, the parameter predictor predicts $\theta_{wb} \in [0,1]^3$ which is rescaled to $\theta'_{wb} \in [\frac{1}{1.1}, 1.1]^3$. These rescaled parameters are then converted into the final per-pixel gain values through a luminance-preserving normalization. With $\theta'_{wb} = (\theta'_{wb,R},\, \theta'_{wb,G},\, \theta'_{wb,B})^\top$, we compute gain values using:

$$\tilde{\theta}'_{wb} = \frac{\theta'_{wb}}{0.27\,\theta'_{wb,R} + 0.67\,\theta'_{wb,G} + 0.06\,\theta'_{wb,B}}. \tag{15}$$

The white balanced pixel is then obtained by applying the diagonal matrix of normalized gains:

$$\mathbf{I}' = \begin{bmatrix} \tilde{\theta}'_{wb,R} & 0 & 0 \\ 0 & \tilde{\theta}'_{wb,G} & 0 \\ 0 & 0 & \tilde{\theta}'_{wb,B} \end{bmatrix} \mathbf{I}. \tag{16}$$

Denoise The parameter predictor predicts a parameter $\theta_d \in [0,1]$, which is mapped to a denoising strength $\theta'_d \in [10^{-5}, 1]$. This value controls the smoothing level of the non-local means (NLM) operator.

For each pixel, the denoised value is computed as a weighted average of neighboring pixels:

$$\mathbf{I}' = \frac{\sum_{y\in\mathcal{N}(x)} w(x,y \mid \theta'_d)\,\mathbf{I}(y)}{\sum_{y\in\mathcal{N}(x)} w(x,y \mid \theta'_d)}, \tag{17}$$

where y indexes spatial locations in the neighborhood $\mathcal{N}(x)$, and $w(x,y \mid \theta'_d)$ denotes the NLM weight between pixel x and pixel y, scaled according to the denoising strength θ'_d.

Sharpen/Blur For a module that applies sharpening or blurring operation, the parameter predictor predicts $\theta_{sh} \in [0,1]$ which is rescaled to $\theta'_{sh} \in [10^{-5}, 10]$. We then define a 3×3 blur kernel K

$$K = \frac{1}{13}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{18}$$

to produce a blurry pixel $\mathbf{I}_{\text{blur}}$ by convolving K with the input image. The final output is then obtained by

$$\mathbf{I}' = \theta'_{sh}\,\mathbf{I} + (1-\theta'_{sh})\,\mathbf{I}_{\text{blur}}. \tag{19}$$

Color correction For color correction, the parameter predictor predicts $\theta_{ccm} \in [0,1]^9$ which is rescaled to $\theta'_{ccm} \in [-2,2]^9$. Then, we define a color correction matrix M with the nine elements of θ'_{ccm} and normalize each row. With the color correction matrix, we apply:

$$\mathbf{I}' = M\,\mathbf{I}. \tag{20}$$

S3.2. Network Architecture Details

S3.2.1. Sequence Predictor

We instantiate the sequence-level policy with a GRU-based autoregressive model that predicts ISP module indices. Let $A = (a_1, \ldots, a_T)$ denote a module sequence, where $T \leq M+1$. Here, M is the number of available ISP modules,

and we additionally include a special end-of-sequence token <eos>. Thus, the module pool contains M actual modules plus one <eos> token, and the sequence length T can be up to $M + 1$ when <eos> appears in the final position. At each recurrent step t, the previously predicted module index a_{t-1} is embedded by a learnable token embedding layer, and passed through a GRU. Then, the GRU produces a hidden state $h_t \in \mathbb{R}^H$ from the embedded token where $H = 128$ in our implementation.

To provide the decoder with an explicit notion of the step index for step-dependent behavior, we additionally use a learnable step embedding s_t for $t \in 1, \ldots, T$ of dimension 16, following FiLM [10]. This step embedding is transformed into FiLM parameters (γ_t, β_t) through a small MLP and is used to modulate the hidden state h_t as

$$\tilde{h}_t = h_t \odot (1 + \gamma_t) + \beta_t, \tag{21}$$

where $\odot$ denotes element-wise multiplication. The modulated hidden state $\tilde{h}_t$ is then projected by a linear decoder to produce the logits over module indices and the <eos> token. To avoid introducing any bias toward specific modules at the beginning of training, we initialize the decoder such that all logits are identical: the weight matrix is set to zero and all bias entries are set to the same constant, so that the initial policy corresponds to a uniform probability distribution over actions after the softmax.

We sample the next action from the resulting probability distribution with a temperature-controlled softmax; during training, the temperature τ follows an exponentially decaying schedule to gradually reduce exploration. To avoid repeated selection of modules, we mask out the already selected modules in the logit space so that each module can be chosen at most once. The decoding process terminates when the <eos> token is selected.

Temperature sampling To balance exploration and exploitation during training, we decay the sampling temperature τ using an exponential schedule. The temperature scales the logits as $\mathrm{softmax}(z/\tau)$, where a higher τ yields a smoother distribution for broader exploration. As training progresses, τ is gradually reduced, producing a sharper distribution that favors high-reward actions. At inference time, exploration is disabled, and the module with the highest probability is selected.

We set $\tau_{\max} = 2.5$ and $\tau_{\min} = 0.2$, and update τ at training step t as

$$\tau(t) = \tau_{\min} + (\tau_{\max} - \tau_{\min}) \exp\left(-\ln 2 \cdot \frac{t}{h}\right),$$

where $h = 3000$ denotes the half-life in training steps. This schedule halves the gap $\tau(t) - \tau_{\min}$ every h steps, providing a smooth shift from exploration to more deterministic behavior as training stabilizes.

S3.2.2. Parameter Predictor

The parameter predictor is implemented as a lightweight encoder-decoder network that maps the input image to a compact latent vector and then to a concatenated parameter vector for all modules. Given an input image $I \in \mathbb{R}^{3\times H\times W}$, we first downsample it to a fixed spatial resolution of 64×64 using adaptive average pooling. The downsampled image is then passed through three convolutional blocks with LeakyReLU activations, resulting in a feature map $F \in \mathbb{R}^{4C\times 8\times 8}$. We apply both global average pooling and global max pooling to F to obtain two separate pooled vectors, and then concatenate them to form an image feature vector of dimension $8C$.

This vector is passed through a two-layer MLP, where the first fully connected layer reduces the dimension to 256 with a ReLU activation, and the second layer projects it to a latent vector $z \in \mathbb{R}^D$, where D is the latent dimension. Finally, a small decoder MLP maps z to the output parameter vector of dimension P that contains continuous parameters for all modules. The parameters corresponding to the selected modules are then retrieved from this vector. We then apply a `tanh` activation and rescale the output from the range $[-1, 1]$ to $[0, 1]$.

S3.3. Details on Running Other Methods

DRL-ISP DRL-ISP [12] is originally implemented to process images with four channels of (R, G_1, G_2, B). For datasets in RGB format with three channels, it duplicates the G channel to construct four channels of (R, G, G, B). Since the datasets we use provide only RGB images, we duplicate the green channel to form a four-channel representation (R, G, G, B) following DRL-ISP. Using these four-channel images, we retrain the proxy networks of DRL-ISP so that they operate correctly on this input format. During ISP pipeline optimization with the proxy networks, we also use the same four-channel (R, G, G, B) images.

ReconfigISP ReconfigISP [15] includes several RAW-domain ISP modules, such as RAW $\rightarrow$ RAW denoising (Bilateral-Bayer, Median-Bayer, NLM-Bayer, Path-Restore-Bayer) and RAW $\rightarrow$ sRGB demosaicking (Laplacian, Nearest, Bilinear, DemosaicNet), all of which are designed for Bayer inputs. We exclude the modules that operate in the RAW domain since our data are entirely in the RGB domain. We then use only the modules that operate in the RGB domain. Moreover, since the original ReconfigISP fixes the number of RGB modules to three, we also set our pipeline length to three for consistency.

AdaptiveISP AdaptiveISP [14] operates under the same input assumptions as our method; therefore, we employ their default configuration without any additional modifications.

S3.4. Details on Dataset

Adobe FiveK Dataset In the Adobe FiveK dataset [2], the RAW images and the expert-retouched images are not geometrically aligned because the experts process the images with Adobe Lightroom, whose internal pipeline introduces subtle geometric transformations (e.g., resampling, slight cropping, or perspective adjustments). Because our training relies on pixel-wise loss computation, the input and target images need to be geometrically aligned. To ensure this alignment, we process the original RAW images through the Adobe Lightroom pipeline and then convert the results to linear RGB. The overall conversion procedure follows exactly the method in [6]. We train the models on 3,000 images and evaluate them on 1,000 test images.

LOD Dataset The LOD dataset [5] provides two subsets: LOD-Normal (well-lit) and LOD-Dark (low-light). For LOD-Dark, we follow the same experimental protocol as prior task-driven ISP works, including AdaptiveISP [14], and use the identical data split. LOD-Dark consists of 1,781 training images and 400 test images. In addition, we evaluate on LOD-All, which combines both Normal and Dark subsets. LOD-All contains 3,122 training images and 1,336 test images, where both the training and test splits include an equal number of Normal and Dark images.

LIS Dataset The LIS dataset [3] provides segmentation annotations for instance-level evaluation. In our experiments, LIS-Dark contains 1,561 training images and 669 test images, and LIS-All includes 3,122 training images and 1,338 test images, each split having an equal number of Normal and Dark images.

KITTI Dataset For monocular depth estimation, we use the KITTI dataset [4]. Following DRL-ISP [12], we adopt their synthetic RAW version of KITTI, which is generated by converting the original RGB images into RAW Bayer images. The resulting dataset consists of 37,430 training images and 697 test images. We apply demosaicing to the synthetic RAW images and use the resulting images as inputs to our ISP optimization framework.

References

[1] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *NeurIPS*, 2019. 1, 4

[2] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011. 9

[3] Linwei Chen, Ying Fu, Kaixuan Wei, Dezhi Zheng, and Felix Heide. Instance segmentation in the dark. *IJCV*, 2023. 4, 9

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 9

[5] Yang Hong, Kaixuan Wei, Linwei Chen, and Ying Fu. Crafting object detection in very low light. In *BMVC*, 2021. 3, 4, 5, 9

[6] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM TOG*, 2018. 9

[7] Adam Jelley, Trevor McInroe, Sam Devlin, and Amos Storkey. Efficient offline reinforcement learning: The critic is critical. *arXiv*, 2024. 3

[8] Mengqi Lei, Siqi Li, Yihong Wu, Han Hu, You Zhou, Xinhu Zheng, Guiguang Ding, Shaoyi Du, Zongze Wu, and Yue Gao. Yolov13: Real-time object detection with hypergraph-enhanced adaptive visual perception. *arXiv*, 2025. 4

[9] Alberto Maria Metelli, Matteo Pirotta, Daniele Calandriello, and Marcello Restelli. Safe policy iteration: A monotonically improving approximate policy iteration approach. *JMLR*, 2021. 3

[10] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv*, 2017. 8

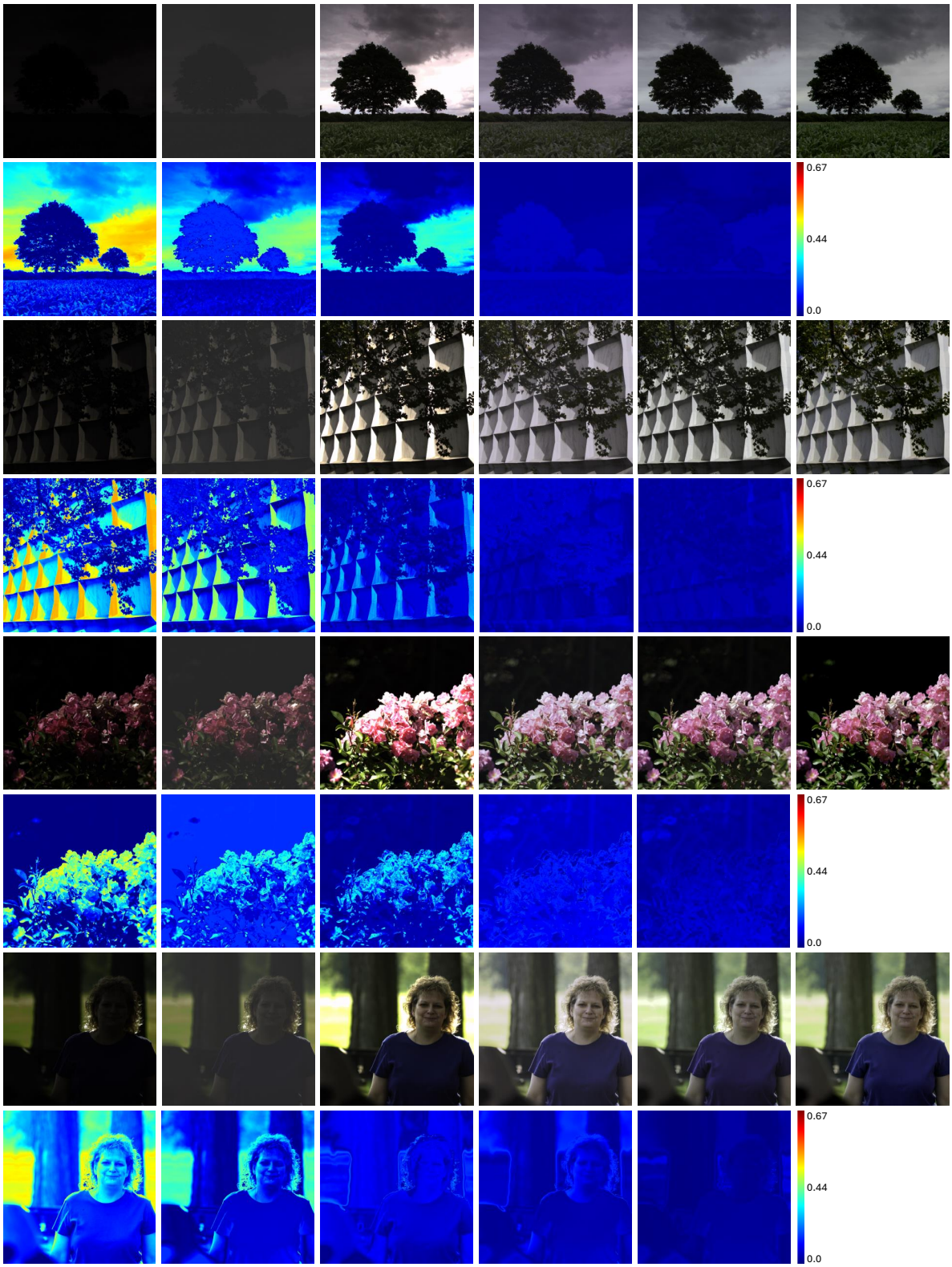
[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 3, 5

[12] Ukcheol Shin, Kyunghyun Lee, and In So Kweon. Drl-isp: Multi-objective camera isp with deep reinforcement learning. In *IROS*, 2022. 1, 2, 3, 4, 8, 9

[13] Paul Wagner. A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. In *NeurIPS*, 2011. 3

[14] Yujin Wang, Tianyi Xu, Fan Zhang, Tianfan Xue, and Jinwei Gu. Adaptiveisp: Learning an adaptive image signal processor for object detection. In *NeurIPS*, 2024. 1, 2, 3, 4, 6, 8, 9

[15] Ke Yu, Zexian Li, Yue Peng, Chen Change Loy, and Jinwei Gu. Reconfigisp: Reconfigurable camera image processing pipeline. In *ICCV*, 2021. 1, 4, 8



(a) Input RAW (b) DRL-ISP (c) ReconfigISP (d) AdaptiveISP (e) POS-ISP (Ours) (f) Ground truth

Figure S5. **Qualitative comparison on the image enhancement task.** We use the images retouched by Expert C from the Adobe FiveK dataset as ground truth. Error maps are also visualized to highlight pixel-wise differences from the ground truth.