Task-Aware Image Signal Processor for Advanced Visual Perception

Supplementary Material

A. Implementation Details

In this section, we provide additional implementation details for our experiments.

Object Detection For object detection experiments on the ROD dataset [32], all ISP models were trained jointly with a pretrained YOLOX-Tiny detector [7]. We randomly selected 90% of the images for training and used the remaining 10% for testing. The batch size was set to 16, and the training images were resized to 640×640. The joint models were trained for 300 epochs using the SGD optimizer with an initial learning rate of 2.5×10^{-3} on two NVIDIA GeForce RTX 4090 GPUs.

For experiments on the PASCAL RAW dataset [23], we adopted RetinaNet [20] with two different backbones, ResNet-18 [11] and ResNet-50. Training was conducted for 50 epochs, with images randomly cropped to a resolution of $(400, 667)$. The dataset was split into training and testing sets according to the official PASCAL RAW protocol [23]. All models were trained on a single NVIDIA GeForce RTX 4090 GPU using the SGD optimizer. The initial learning rate was set to 5×10^{-3} for the ResNet-18 backbone and 2×10^{-3} for the ResNet-50 backbone.

For experiments on the LOD dataset [13], we employed RetinaNet with a ResNet-50 backbone. The batch size was set to 4, and the models were trained for 35 epochs using the SGD optimizer with an initial learning rate of 5×10^{-3}. All training was performed on a single NVIDIA GeForce RTX 4090 GPU.

Segmentation For semantic segmentation experiments on the synthetic ADE20K dataset [36], we followed the settings of Cui *et al.* [4]. Training images were cropped to 512×512, and the number of training iterations was set to 80,000. The models were trained on four NVIDIA GeForce RTX 4090 GPUs with Adam optimizer.

B. Traditional Image Signal Processor

In a conventional image signal processing (ISP) pipeline, raw sensor measurements are transformed into display-ready RGB images through a sequence of modular operations. Below we summarize the typical modules and their primary functions, presented in an approximate processing order.

1. **Raw preprocessing.** Raw sensor data are first corrected for sensor bias by subtracting the black level and optionally applying gain/offset corrections. Linearization compensates for non-linear ADC responses so subsequent operations operate in a scene-linear radiometric domain.
2. **Defective-pixel correction and hot-pixel removal.** Pixels known to be dead or noisy are detected and replaced, typically by interpolation from neighboring pixels or median filtering, to prevent localized artifacts from propagating through the pipeline.
3. **Demosaicing.** Demosaicing reconstructs full RGB color at each pixel from the mosaiced sensor pattern (e.g., Bayer). Algorithms estimate missing color components while aiming to preserve edges and minimize color zippering or bleeding.
4. **Denoising.** Denoising reduces photon and readout noise present in the linear raw domain. Effective denoising preserves fine texture and edges while suppressing stochastic noise, often using spatial, temporal (for burst), or frequency-domain filtering.
5. **White balance.** White balance scales the three color channels to compensate for the scene's illumination color so that neutral surfaces appear neutral. This operation is typically performed in the linear domain and may be driven by metadata, statistics, or automatic estimation.
6. **Color correction.** A color correction matrix maps camera-native RGB to the target colorimetry (e.g., linear scene-referred RGB or display primaries). The 3×3 matrix corrects systematic spectral differences between sensor responses and the desired color space.
7. **Tone mapping and dynamic range compression.** Tone mapping converts scene-referred linear intensities to display-referred values, compressing high dynamic range to the limited display range while preserving perceptual contrast. This step may use parametric curves or more complex mapping functions.
8. **Gamma correction.** Nonlinear gamma encoding (e.g., the sRGB transfer) is applied to map linear scene luminance into a perceptually uniform display-referred space, improving visual fidelity on standard displays.
9. **Sharpening and detail enhancement.** Sharpening boosts high-frequency components to improve perceived crispness. Methods typically apply edge-aware unsharp masking or high-frequency residuals while mitigating halo artifacts.
10. **Compression and encoding.** Finally, the rendered RGB images are quantized and encoded (e.g., JPEG, HEIF) for storage or transmission, including any chroma subsampling and metadata embedding.

Number	PASCAL RAW	LOD
n=4	88.6	62.2
n=8	89.0	63.3
n=16	89.9	63.9
n=32	88.4	62.0

Table 7. Ablation on the number of mask layers on PASCAL RAW dataset [23] and LOD dataset [13].

C. Ablation on the Number of Mask Layers

We conducted experiments on the PASCAL RAW dataset [23] using RetinaNet with ResNet-18 and on the LOD dataset [13] using RetinaNet with ResNet-50 to determine the optimal number of mask layers. Specifically, we evaluated models with 4, 8, 16, and 32 layers, as summarized in Table 7. The results show that increasing the number of mask layers generally improves performance on both daytime and nighttime datasets. However, when the number of layers becomes too large, performance drops significantly. Based on these observations, we set the number of mask layers to 16 in this work.

D. More Comparison with State-of-the-Art

To further demonstrate the superiority of our method, we compare it with a recent state-of-the-art approach, DarkISP [8]. We follow the experimental settings of DarkISP, where the original RAW images are directly fed into the ISP network without performing demosaicking, and the model is jointly trained with RetinaNet [20] using a ResNet-18 backbone [11] on the LOD dataset [13]. The results are reported in Table 8. Our method achieves better performance while maintaining more than 10$\times$ faster inference speed.

Dataset	Model	Method	mAP	Time (ms)
LOD	RetinaNet + ResNet-18	Dark-ISP	64.9	430.41
		TA-ISP (Ours)	**66.0**	**26.43**

Table 8. Comparison with DarkISP [8] on the LOD dataset [13].

E. More Visualization Results

In this section, we present additional visualization results. We compare our method with Demosaic, InvISP [31], MW-ISPNet [14], DIAP [32], and RAW-Adapter [4]. The detection results are shown in Figure 5, where rows 1–4 correspond to the PASCAL RAW dataset [23], rows 5–7 correspond to the LOD dataset [13], and rows 8–9 correspond to the ROD dataset [32]. The segmentation results are illustrated in Figure 6, where we provide a side-by-side comparison with the aforementioned methods.

Figure 5. Visual results on PASCAL RAW dataset [23] , LOD dataset [13] and ROD dataset [32]. (Zoom in for best view).

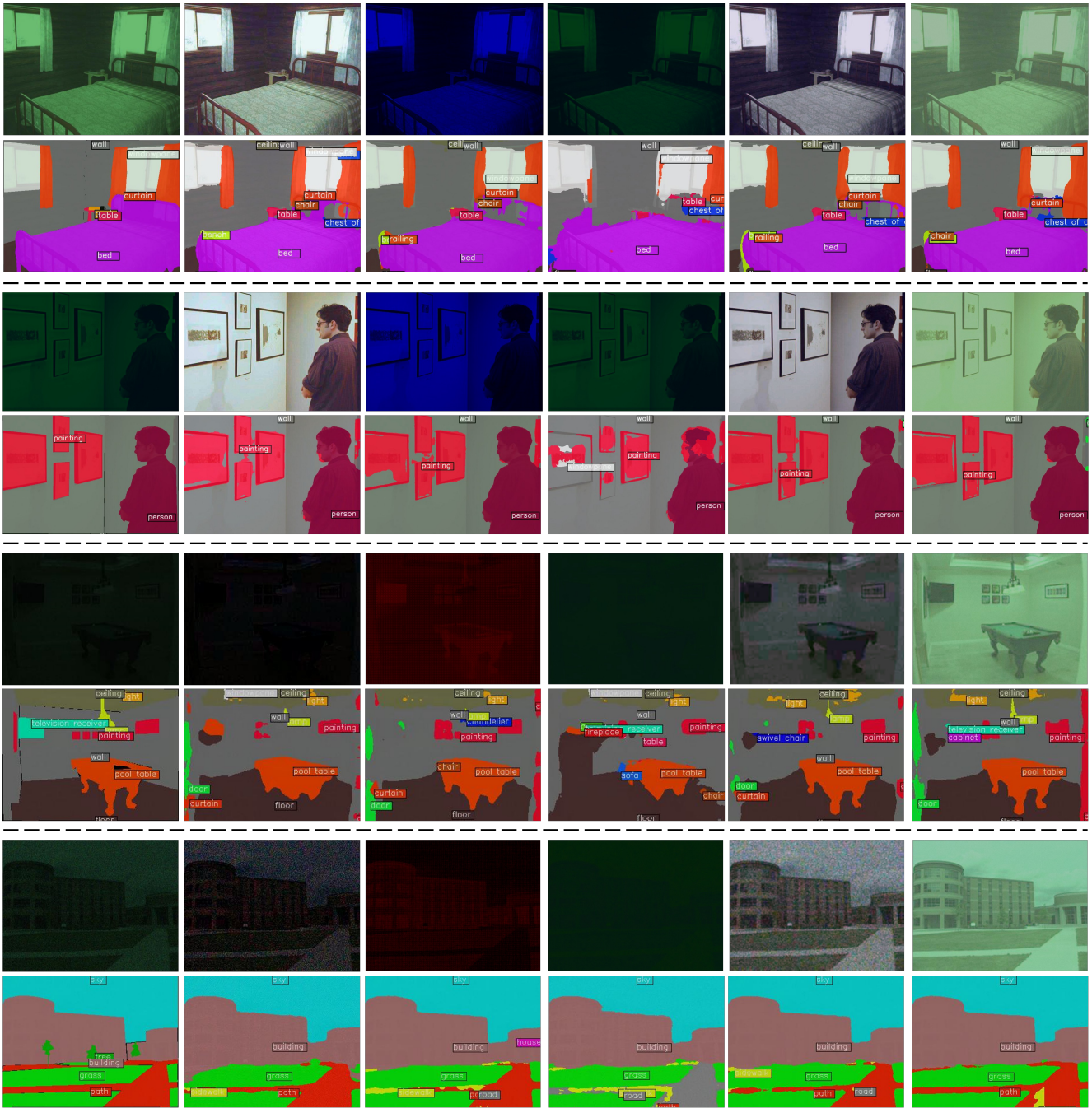

Figure 6. Semantic segmentation results on normal and low-light RAW data on ADE20K dataset [36]. (Zoom in for best view).