Torch-ISP:GPU Accelerated PyTorch-based Camera Image Signal Processing (ISP)

Student Name: Kaustubh Sadekar
Instructor: Feng Liu

Abstract

This project details the development of a custom Image Signal Processor (ISP) pipeline built using PyTorch, with a strong emphasis on vectorized computations and GPU acceleration. The primary goal is to replicate and optimize the fundamental steps involved in converting raw sensor data into a visually pleasing RGB image, traditionally handled by dedicated hardware or libraries like `rawpy`. The report outlines the implementation of core ISP components, their integration into a coherent pipeline, and performance comparisons against conventional methods.

1 Introduction

This project focuses on building an Image Signal Processor (ISP) using PyTorch, with a strong emphasis on vectorized operations and GPU acceleration. The pipeline aims to convert raw DNG sensor data into a processed RGB image, demonstrating a software-based alternative to conventional ISP hardware or libraries. The motivation stems from the desire to create a flexible, efficient, and programmable ISP that leverages modern GPU architectures for real-time processing. This allows for customization of each stage and opens avenues for integration with deep learning models.

2 Architecture and Key Components

The PyTorch-based ISP pipeline processes raw DNG image data through a series of sequential and modular steps, each implemented as a PyTorch function for efficient GPU processing. The overall flow is as follows:

Raw DNG Image Data
↓
Black Level Compensation
↓
Mosaic to GBRG Stack Conversion
↓
Automatic White Balance (AWB)
↓
Demosaicing
↓
Denoising and Sharpening
↓
Processed RGB Image

2.1 Key Implementation Details

- **Black Level Compensation (`blacl_level_comp`):** Subtracts per-channel black level values from the raw sensor data and clips negative results to zero.
- **Raw Image Restructuring (`mosaic_to_gbrg_stack`):** Converts the mosaiced Bayer pattern into a 4-channel GBRG stack, simplifying subsequent demosaicing. This assumes a '|GB| /RG/' CFA pattern.
- **Automatic White Balance (`gbrg_stack_awb`):** Applies a simple gray-world assumption, calculating gain factors for red and blue channels based on the mean of the green channels.
- **Demosaicing (`demosaic_gbrg`):** Transforms the 4-channel GBRG stack into a 3-channel RGB image using bilinear interpolation via `torch.nn.functional.interpolate`. Green channel reconstruction involves averaging interpolated green components.
- **Denoising and Sharpening (`denoise_rgb`):** Utilizes 2D convolutional filters (`F.conv2d`). Denoising is achieved through Gaussian blurring, and sharpening is optionally applied by subtracting a Laplacian filter's output from the blurred image.

3 Experimental Results and Analysis

The ISP pipeline was tested using a DNG image (`img1.dng`), with performance evaluated based on visual quality and execution time on both CPU and GPU.

3.1 Performance Comparison

The custom PyTorch ISP, particularly when accelerated by a GPU, demonstrates significant speed advantages. A comparison of processing times shows the efficiency gained:

Pipeline Stage	**Device**	**Time (s)**
Rawpy Postprocess (Linear Demosaic, Auto WB)	CPU	0.805
Full ISP Pipeline (BLC, GBRG Stack, AWB, Demosaic, Denoise/Sharpen)	CUDA	**0.0094**

The GPU-accelerated PyTorch ISP is orders of magnitude faster than `rawpy`'s CPU-based post-processing for a comparable set of operations.

3.2 Visual Quality Notes

While the custom ISP output still exhibits a small mean difference (e.g., -0.011) compared to `rawpy`'s, it successfully replicates the core functionalities. The effectiveness of denoising and sharpening was visually apparent in observations.

4 Discussion, Limitations, and Future Work

The project successfully demonstrated a vectorized, GPU-accelerated ISP pipeline in PyTorch, highlighting its speed and modularity. The GPU acceleration provides a significant performance boost compared to CPU-bound libraries.

4.1 Limitations

- **Algorithmic Simplicity:** The AWB and demosaicing algorithms are relatively basic (gray-world AWB, bilinear demosaicing).
- **Fixed Parameters:** Denoising/sharpening parameters are static, lacking adaptability for diverse image content.
- **Incomplete Pipeline:** Lacks critical post-processing steps like tone mapping, gamma correction, and detailed color management for display-ready images.

4.2 Future Work

Future enhancements could involve integrating more advanced demosaicing (e.g., AHD, VNG), adaptive denoising techniques (possibly AI-based), comprehensive tone mapping, and precise color correction. Expanding the dataset for evaluation and exploring end-to-end differentiable ISP architectures represent promising avenues for further research and development.

5 Conclusion

This project validates the feasibility and performance advantages of building an ISP pipeline using PyTorch for GPU-accelerated processing. By providing a flexible and efficient software-based ISP, this work serves as a strong foundation for exploring advanced computational photography techniques and integrating deep learning methodologies into image signal processing workflows.