# Image Signal Processing with CUDA

Ching-Yang Lin
National Yang Ming Chiao Tung University Hsinchu, Taiwan
linjohnss.cs12@nycu.edu.tw

Ying-Huan Chen
National Yang Ming Chiao Tung University Hsinchu, Taiwan
yinghuanchen.cs13@nycu.edu.tw

Chung-Ho Wu
National Yang Ming Chiao Tung University Hsinchu, Taiwan
chunghowu.cs12@nycu.edu.tw

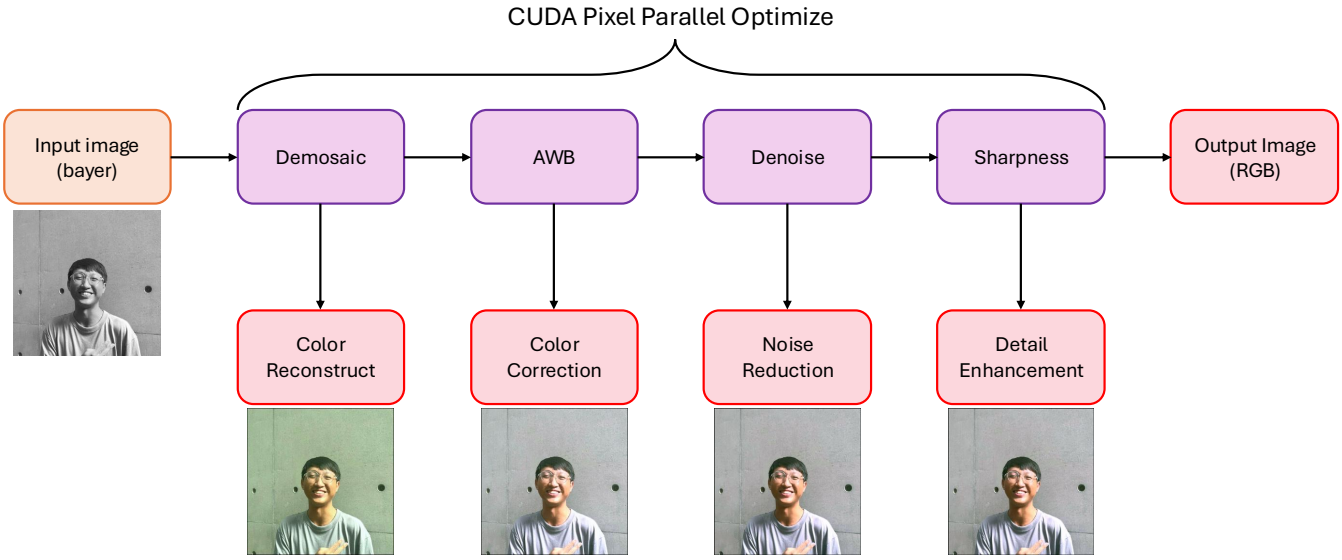


**Figure 1: Block Diagram of Image Signal Processign with CUDA.**

## Abstract

This project implements a CUDA-accelerated Image Signal Processing (ISP) pipeline to optimize image processing performance on GPU hardware. The pipeline includes key image processing stages: demosaicing, auto white balance (AWB), denoising, and sharpness enhancement. By leveraging NVIDIA's CUDA framework, we achieved significant speedup compared to sequential CPU processing across different image sizes and kernel configurations. Experimental results demonstrate that the GPU implementation achieves up to 5.47x speedup for large images and 15.1x speedup for complex operations with 7x7 kernels. The pixel-parallel approach enables efficient processing of high-resolution images while maintaining image quality. Performance analysis shows that the speedup scales positively with both image size and kernel complexity, validating the effectiveness of our GPU-accelerated solution.


*Conference acronym 'XX, June 03–05, 2018, Woodstock, NY*

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

**ACM Reference Format:**
Ching-Yang Lin, Ying-Huan Chen, and Chung-Ho Wu. 2018. Image Signal Processing with CUDA. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

With the advancement of camera technology and the widespread adoption of multimedia applications in robotics, self-driving cars, and AR/VR systems, Image Signal Processors (ISPs) have become a critical component in modern digital image processing. ISPs handle raw image data captured by camera sensors and perform a series of essential tasks: demosaicing for full color reconstruction, auto white balance for color accuracy, noise reduction for image clarity, and sharpness enhancement for detail preservation. Traditionally, ISPs rely on dedicated hardware to accelerate these operations. However, as the demand for real-time processing in high-resolution imaging increases, the rigidity of specialized hardware proves inadequate in meeting the needs of emerging applications.

CUDA technology offers the capability to perform highly parallel computations on GPUs, providing greater computational power and flexibility compared to traditional CPUs or dedicated ISP hardware. By implementing the ISP processing pipeline in CUDA, we demonstrate significant performance improvements, particularly in computationally intensive operations with larger kernel sizes and higher resolution images. Our experiments show that GPU acceleration achieves up to 5.47x speedup for 12000x8000 pixel images

and 15.1x speedup for 7x7 kernel operations, while maintaining high image quality throughout the pipeline stages.

The parallel nature of ISP operations, such as convolution-based denoising and sharpness enhancement, makes them particularly well-suited for GPU acceleration. Each pixel in these operations can be processed independently, allowing for efficient parallelization across thousands of CUDA cores. Our implementation leverages this architectural advantage by optimizing memory access patterns and computational resources for each pipeline stage. For instance, our denoising and sharpness enhancement stages show particularly impressive speedups due to their regular memory access patterns and high arithmetic intensity, which align well with GPU's parallel processing capabilities.

Moreover, the flexibility of a software-based CUDA implementation allows for easy modification and experimentation with different processing algorithms. This adaptability is crucial for modern imaging applications where processing requirements may vary based on scene conditions or application-specific needs. Our qualitative results demonstrate that the GPU-accelerated pipeline maintains excellent image quality while significantly reducing processing time, making it suitable for real-time applications in robotics and autonomous systems.

The results of our implementation demonstrate that CUDA-based ISPs offer a promising solution for applications requiring both high performance and flexibility in image processing pipelines. As imaging technology continues to advance toward higher resolutions and more complex processing requirements, the scalable nature of GPU acceleration becomes increasingly valuable. This work provides a foundation for future development of efficient, flexible, and high-performance image processing systems.

## 2 Proposed Solution

This project utilizes CUDA acceleration to implement an image signal processing (ISP) system, aiming to enhance the efficiency of image processing. As shown in the system architecture, the main processing flow includes the input of raw Bayer format images, which undergo multiple steps before generating the final RGB format output images. Below is a description of the function of each module and our parallelized method.

### 2.1 Demosaic

In our simplified ISP pipeline, we process Bayer format (RGGB) images directly from the camera sensor. Demosaic is a crucial step in converting these Bayer format images into full RGB color images. This process uses interpolation to reconstruct complete red, green, and blue values for each pixel from the original monochrome array. The interpolation method can be expressed by the equation:

$$C_{missing} = \frac{\sum_{i=1}^{n} C_i}{n} \tag{1}$$

where C represents the color channel (R, G, or B), and n is the number of adjacent pixels of the same color used in the calculation. Since demosaic requires computing values based on neighboring pixels, it significantly benefits from CUDA's parallel computing capabilities, especially when processing high-resolution images. We leverage CUDA to parallelize these interpolation operations, allowing multiple pixel calculations to be performed simultaneously, rather than processing them sequentially. This parallel approach substantially improves the computational efficiency of our pipeline."

### 2.2 Auto White Balance (AWB)

Due to the inherent nature of Bayer format images, the overall color tone tends to be greenish after demosaicing. The AWB (Auto White Balance) module addresses this issue by automatically adjusting the white balance. This process dynamically modifies the weights of the red, green, and blue channels based on the image's color distribution, resulting in more natural color representation.

Our implementation employs the Gray-World AWB algorithm, which operates under the assumption that the average intensities of RGB channels should be approximately equal. The algorithm first calculates the average values for each RGB channel separately, then computes scaling factors to align the Red and Blue channels with the Green channel, which can be expressed as follows:

$$\mu_r = \frac{1}{M \cdot N} \sum_{x=1}^{M} \sum_{y=1}^{N} I_r(x, y), \tag{2}$$

$$\mu_g = \frac{1}{M \cdot N} \sum_{x=1}^{M} \sum_{y=1}^{N} I_g(x, y), \tag{3}$$

$$\mu_b = \frac{1}{M \cdot N} \sum_{x=1}^{M} \sum_{y=1}^{N} I_b(x, y), \tag{4}$$

$$\hat{\alpha}_r = \frac{\mu_g}{\mu_r}, \quad \hat{\alpha}_b = \frac{\mu_g}{\mu_b} \tag{5}$$

$$\hat{I}_r(x, y) = \hat{\alpha}_r I_r(x, y), \quad \hat{I}_b(x, y) = \hat{\alpha}_b I_b(x, y) \tag{6}$$

Given that this process involves handling a large number of pixels, our CUDA-accelerated approach parallelizes two key steps: first, the average calculation in Equation (1), and then the pixel-wise scaling in Equation (3). This parallel processing ensures efficient real-time adjustments, resulting in RGB images that more accurately reflect real-world scene colors.

### 2.3 Denoise

The denoise module is introduced after the AWB process to remove any noise present in the RGB image. Denoising is a critical step, especially in high-resolution images where noise can degrade image quality. Since noise typically manifests as outlier values among neighboring pixels, we implement a median filter in our pipeline. The median filter can be expressed as:

$$y[m, n] = \text{median} x[i, j], (i, j) \in w \tag{7}$$

where $y[m, n]$ is the output pixel value, $x[i, j]$ represents the input pixel values within a window $w$ centered at position $(m, n)$. This filter effectively removes spurious pixel values while preserving important edge details in the image. We leverage CUDA to implement this denoising algorithm with parallel processing capabilities, allowing simultaneous operations on multiple pixels. This parallel approach ensures efficient noise reduction while maintaining image detail quality, producing a cleaner and clearer image for subsequent processing stages.

| Image size | Demosaic | AWB | Denoise | Sharpness | Total | Speedup |
|---|---|---|---|---|---|---|
| 3000 x 2000 | 0.131054 | 0.056723 | 0.531934 | 0.417839 | 1.13755 | |
| | 0.328342 | 0.020322 | 0.083831 | 0.066165 | 0.597162 | 1.9 |
| 6000 x 4000 | 0.528682 | 0.226113 | 2.101337 | 1.682355 | 4.538487 | |
| | 0.305317 | 0.080259 | 0.352535 | 0.260054 | 0.998165 | 4.55 |
| 12000 x 8000 | 2.107046 | 0.887871 | 8.526944 | 6.654999 | 18.17686 | |
| | 0.558354 | 0.299112 | 1.447902 | 1.015010 | 3.320378 | 5.47 |

**Table 1: Performance comparison with different image sizes (time in seconds)**

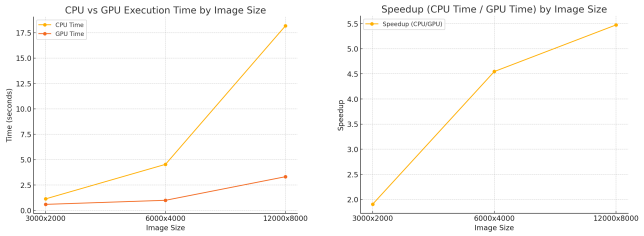


**Figure 2: CPU vs GPU execution time comparison as kernel size increases (left) and the corresponding speedup ratio (right)**

## 2.4 Sharpness

After denoising, the sharpness enhancement module is applied to improve image detail clarity. Sharpness enhancement is particularly important for high-resolution images where edge definition can impact overall image quality. The process can be described by the Laplacian-based sharpening equation:

$$E[m,n] = \sum_{i,j \in w} L[i,j] \cdot I[m+i, n+j] \tag{8}$$

where $L[i,j]$ represents the Laplacian kernel:

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{9}$$

The final sharpened image is obtained by:

$$I_{sharp}[m,n] = I[m,n] + \alpha \cdot E[m,n] \tag{10}$$

where $\alpha$ is the sharpening strength factor, $I[m,n]$ is the original image, and $E[m,n]$ represents the detected edge details. To optimize performance, our CUDA implementation follows a two-stage parallel processing approach at the pixel level:

First, we parallelize the edge detection computation by assigning each pixel to a separate CUDA thread, allowing simultaneous Laplacian filter operations across the entire image Then, we parallelize the final sharpening step by having each CUDA thread handle the addition of edge details to the corresponding original pixel

This pixel-level parallel processing strategy enables efficient real-time sharpness enhancement while preserving important image details.

# 3 Experimental Methodology

Our experimental evaluation was conducted on a high-performance computing platform featuring an AMD Ryzen Threadripper PRO processor paired with an NVIDIA GeForce RTX 4090 GPU equipped with 24GB memory, running CUDA 11.7. We designed our experiments to assess both the scalability and computational efficiency of our CUDA-ISP implementation compared to sequential CPU processing.

To comprehensively evaluate performance scaling with image size, we tested our implementation across three different resolutions: 3000x2000, 6000x4000, and 12000x8000 pixels. This range allows us to understand how our parallel implementation performs from moderate to high-resolution image processing scenarios. Additionally, we investigated the impact of computational complexity by varying convolution kernel sizes from 3x3 to 5x5 and 7x7, as kernel operations are fundamental to several stages in our pipeline.

Throughout our testing, we measured execution times for each stage of our ISP pipeline including demosaicing, auto white balance, denoising, and sharpness enhancement. By comparing the processing times between sequential CPU implementation and our CUDA-based parallel approach, we were able to calculate meaningful speedup metrics. We maintained careful documentation of both timing measurements and visual outputs at each pipeline stage to ensure that performance improvements did not come at the cost of image quality.

# 4 Experimental Results

## 4.1 Quantitative Results

We evaluated our CUDA-ISP implementation against CPU sequential processing across different image sizes and kernel configurations. For image size scaling experiments, we tested three resolutions: 3000x2000, 6000x4000, and 12000x8000 pixels. As shown in table 1, the CUDA implementation achieved consistent speedup across all stages of the ISP pipeline. The speedup factor increases with image size, from 1.9x for 3000x2000 images to 5.47x for 12000x8000 images. This scaling behavior is attributed to better GPU thread utilization with larger images, as the overhead of kernel launch and memory transfer becomes relatively insignificant compared to the actual computation time. fig. 2 illustrates this scaling trend, where the GPU execution time grows much more slowly than CPU time as image size increases.

The impact of kernel size was also investigated using 3x3, 5x5, and 7x7 convolution kernels. As demonstrated in table 2 , larger kernel sizes lead to more significant performance gains. The speedup ranges from 6.34x for 3x3 kernels to 15.1x for 7x7 kernels. This substantial improvement with larger kernels can be attributed to two factors: first, larger kernels increase arithmetic intensity (computations per memory access), allowing better hiding of memory

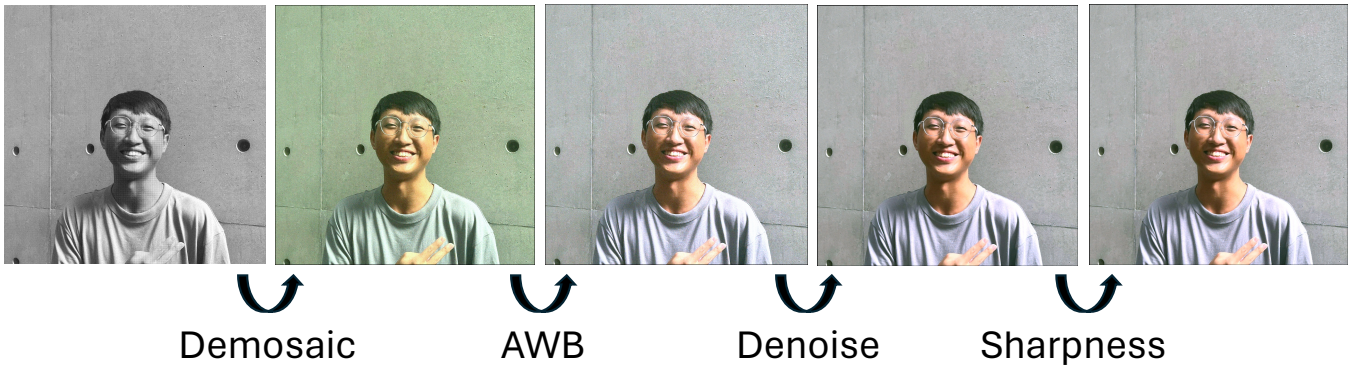


**Figure 3: Visual demonstration of our ISP pipeline stages: starting from raw sensor data, the image undergoes demosaicing to reconstruct full RGB colors, followed by auto white balance correction to adjust color temperature, denoising to remove sensor noise, and finally sharpness enhancement to improve edge details.**

latency; second, the GPU's massive parallel architecture can efficiently distribute the increased workload across multiple streaming multiprocessors (SMs).

Notably, the demosaicing stage shows relatively lower speedup compared to other stages. This is primarily due to its memory access pattern - demosaicing requires irregular neighboring pixel access for interpolation, leading to non-coalesced memory access patterns that cannot fully utilize GPU's memory bandwidth. In contrast, operations like denoising and sharpness enhancement exhibit better performance as they involve regular, structured memory access patterns that can be efficiently coalesced on GPU.

The performance analysis shows that our CUDA implementation is particularly effective for high-resolution images and computationally intensive operations with large kernels. The most significant speedup is observed in the denoising and sharpness enhancement stages, which involve intensive convolution operations with regular memory access patterns. This aligns with GPU architectural strengths in handling parallel workloads with high computational density and regular memory access patterns.

| Kernel size | Total Time | Speedup |
|---|---|---|
| 3x3 | 0.531934 | |
| | 0.083831 | 6.34 |
| 5x5 | 5.884479 | |
| | 0.658415 | 8.94 |
| 7x7 | 12.155817 | |
| | 0.805294 | 15.1 |

**Table 2: Performance comparison with different kernel sizes (time in seconds)**

## 4.2 Qualitative Results

To validate the quality of our CUDA-ISP implementation, we conducted qualitative analysis of the output images at each pipeline stage, as shown in fig. 3. The demosaicing stage effectively reconstructs the full RGB color information from the Bayer pattern,

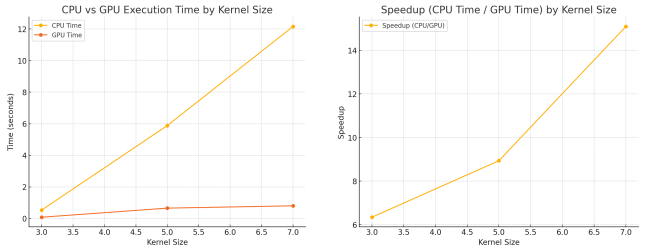


**Figure 4: CPU vs GPU execution time comparison across different image resolutions (left) and the corresponding speedup ratio (right)**

maintaining good color fidelity without introducing significant artifacts. In the AWB stage, we can observe the correction of color cast, particularly in adjusting the greenish tint to achieve more natural color reproduction of the scene and skin tones. The denoising stage demonstrates effective noise reduction while preserving important image details. This is particularly noticeable in the smooth wall areas where noise is suppressed without introducing blur to the subject's facial features and glasses. The final sharpness enhancement stage successfully improves edge definition and local contrast, as evident in the enhanced details of the shirt texture and facial features, without introducing noticeable halos or artifacts. Overall, the visual results confirm that our GPU-accelerated implementation maintains high image quality throughout the pipeline while achieving significant performance improvements. The final output image exhibits natural colors, clean details, and appropriate sharpness, meeting the quality standards expected in modern image processing applications. Additionally, we also provide more visual results in fig. 5.

# 5 Related Works

## 5.1 Image Signal Processing

Image Signal Processing (ISP) is a critical component in modern digital camera systems, responsible for transforming raw sensor data

**Figure 5: More visual results**

into a viewable image. Over the years, significant advancements have been made in ISP techniques to enhance image quality, particularly in noise reduction, demosaicing, white balance, and color correction. Traditional ISP pipelines are implemented on dedicated hardware such as Digital Signal Processors (DSPs) or embedded processors, offering real-time performance but limited flexibility.

However, our approach leverages CUDA acceleration for software-based ISP processing, making it highly adaptable for a wide range of contemporary applications. This software-defined ISP solution is particularly valuable in emerging fields such as robotics, autonomous vehicles, and AR/VR headsets, where computational flexibility and integration with existing GPU-based systems are crucial advantages. The GPU-accelerated approach allows for real-time processing while maintaining the adaptability needed for diverse imaging applications across these modern platforms.

## 5.2 GPU-Accerlerated Image Processing

The use of GPUs for accelerating ISP tasks has garnered considerable attention due to the parallel nature of both ISP and GPU architectures. Early works [2] demonstrated the advantages of utilizing GPUs for real-time image processing, showcasing performance gains in tasks such as denoising and filtering. Since then, frameworks like OpenCV and CUDA have been extensively used to parallelize image processing pipelines. OpenCV offers GPU-accelerated functions for common image manipulation tasks, but is often limited by abstraction layers that prevent optimal utilization of GPU resources for more complex pipelines. CUDA, introduced by NVIDIA, enables fine-grained parallelism and optimization on NVIDIA GPUs, providing substantial speed-ups in various domains, including computer vision and image processing. For example, ImageConvolutionWithCUDA [1] leveraged CUDA to parallelize computationally expensive image processing tasks like convolution, achieving significant performance improvements. However, their work focused on specific tasks without addressing the complete ISP pipeline.

# 6 Conclusion

In this work, we presented a CUDA-based implementation of a complete ISP pipeline demonstrating significant performance improvements over traditional CPU processing. Through parallel optimization of key processing stages - demosaicing, auto white balance, denoising, and sharpness enhancement - we achieved substantial speedups that scale with both image resolution and computational complexity. Our experimental results show considerable performance gains with larger images and kernel sizes while maintaining high image quality across various scenarios. The implementation particularly excels in operations with regular memory access patterns, and future optimizations could explore channel parallelization and kernel-level improvements to further enhance performance for real-time imaging applications.

# References

[1] Victor Podlozhnyuk. 2007. Image convolution with CUDA. *NVIDIA Corporation White Paper* 2097 (07 2007).

[2] Nan Zhang, Yun shan Chen, and Jian li Wang. 2010. Image parallel processing based on GPU. In *2010 2nd International Conference on Advanced Computer Control*, Vol. 3. 367–370. https://doi.org/10.1109/ICACC.2010.5486836