GPU Accelerated Image Processing Technology: Architectural Features, Parallel Optimization and Deep Learning Applications

Shaowei Chen

School of Physics and Information Engineering, Fuzhou University, Fuzhou, 350100, China
172109076@fzu.edu.cn

Abstract. This paper systematically discusses the technical basis and optimization methods of GPU-accelerated image processing, focusing on the analysis of GPU architectural features and CUDA and OpenCL parallel computing platforms. It also covers GPU optimization methods in image defogging and enhancement, frequency domain transform (FFT, DCT), feature detection (SIFT) and deep learning driven image processing. The experimental data shows that the parallel optimization strategy based on GPU, in which the computational efficiency of the defogging algorithm is improved by 10 times, the performance of FFT reaches 1000GFlops, and the acceleration ratio of SIFT is up to 121.99 times. Further combining with deep learning scenarios, GPU-based CNN training optimization improves the GEMM operation throughput by 1.97 times. Furthermore, it constructs the OUR-GAN framework to achieve 16k image generation while reducing memory usage to 12.5GB. The study proves that GPU-based software and hardware co-optimization can significantly improve the efficiency of image processing and provide technical support for real-time applications and large-scale computation.

Keywords: GPU, Image Processing, CUDA, CNN.

1 Introduction

With the rapid development of computer vision and image processing technology, the acquisition and processing of massive image data have become more and more important. Under the traditional CPU architecture, image processing tasks require a large amount of computational resources and time, which makes it difficult to meet the real-time and high-efficiency requirements. Especially in some complex image processing tasks, such as convolution, transformation, and feature extraction operations, the processing power of the CPU appears to be insufficient. For this reason, the GPU has become a key technology for image processing acceleration by virtue of its high concurrent processing capability, multi-core architecture, and powerful floating-point computing capability.

In recent years, GPU acceleration technology has made significant progress in the fields of image defogging, image enhancement, frequency domain transformation (e.g.,

J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_40

FFT and DCT), feature extraction (e.g., SIFT), and deep learning. Through GPU parallel computing, the processing efficiency is increased by tens of times compared to traditional serial processing methods, significantly reducing image processing time. In the field of deep learning, the application of GPUs is particularly prominent; generative adversarial networks (GANs) and convolutional neural networks (CNNs) and other deep learning models accelerated by GPUs have made significant breakthroughs in training efficiency and image generation quality.

However, although GPU acceleration technology has been widely used in several fields, how to further optimize its hardware architecture utilization, memory access strategy, and thread scheduling mechanism is still an important research topic. In particular, when processing large-scale image data, there are still many challenges to maximize the computational potential of GPUs and improve resource utilization. Therefore, the research of this paper focuses on the application of GPU-accelerated image processing techniques and optimization methods.

First, this paper will introduce the architectural characteristics and programming model of GPU to lay the foundation for understanding GPU parallel computing. Then, the specific applications of GPU in image defogging, FFT/DCT transform, feature detection (e.g., SIFT), and deep learning (CNN, GAN) will be analyzed in depth, and the optimization strategies will be discussed. This paper will also demonstrate the effect of GPU optimization techniques in different image processing tasks through experimental data, focusing on the advantages of performance enhancement, resource utilization, and computational efficiency. Ultimately, this paper aims to provide practical guidance for researchers and engineers to promote the wide application of GPUs in image processing and to propose feasible suggestions and ideas for further optimization of GPU architectures. By systematically investigating GPU parallel optimization strategies, this paper not only provides new perspectives for academia but also provides valuable references for engineering practices in industry.

2 GPU accelerated image processing

2.1 GPU architecture characteristics and programming model

The hardware architecture of GPU is composed of multiple parallel processing units and storage structures at different levels, aiming to efficiently handle large-scale data computing tasks. Its core components include the following parts: Graphics processing core: As the core computing unit of the GPU, it is responsible for performing various computing tasks, including graphics rendering and general computing functions. Storage system: The storage unit of the GPU is used for data storage and access, mainly including multi-level storage structures such as global storage, shared storage and register files, which are managed by a memory controller to optimize data access and transmission efficiency. Task Scheduler: used to coordinate and allocate computing tasks, optimize the utilization rate of computing cores, and improve parallel computing efficiency. Input and output interface (I/O interface): Responsible for communicating with the host system, including data transmission and command control functions. These components work together to form the high-performance computing power of

the GPU, enabling it to efficiently handle complex parallel computing tasks [1]. The GPU architecture, as shown in Fig. 1.

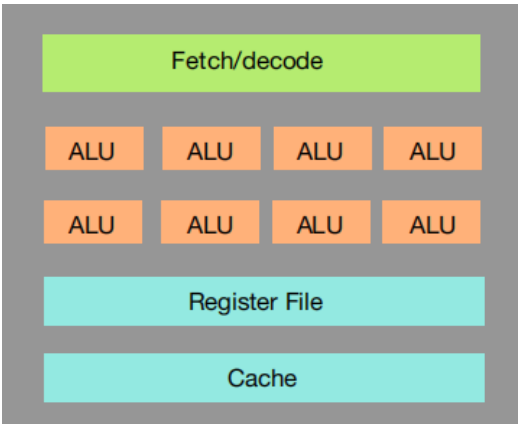

Fig. 1. GPU architecture [1]

2.2 Programming Model

2.2.1 CUDA hardware system.

CUDA is an interface developed by Nvidiada for CPU programming specifically for N cards. CUDA is a popular platform in GPU programming model. CUDA is mainly used in the fields of image reconstruction, image enhancement, image segmentation, and image matching and image classification in GPU [2]. GPU parallel acceleration using the CUDA platform, developers need to understand their hardware system. The core structure of the CUDA device is composed of a series of scalable streaming multiprocessors. The CUDA execution model relies on kernel functions. When executing these functions, compute resources are allocated in the form of thread blocks. Each SM is able to process threads in multiple thread blocks simultaneously, and each thread is responsible for processing one or more data units. This parallel computing model reflects the SIMT execution characteristics of CUDA [3]. The CUDA execution model, as shown in Fig. 2. It can mask the memory access delay through a large number of threads that execute in parallel, improve operational efficiency, and demonstrate excellent performance in the field of graphics rendering.

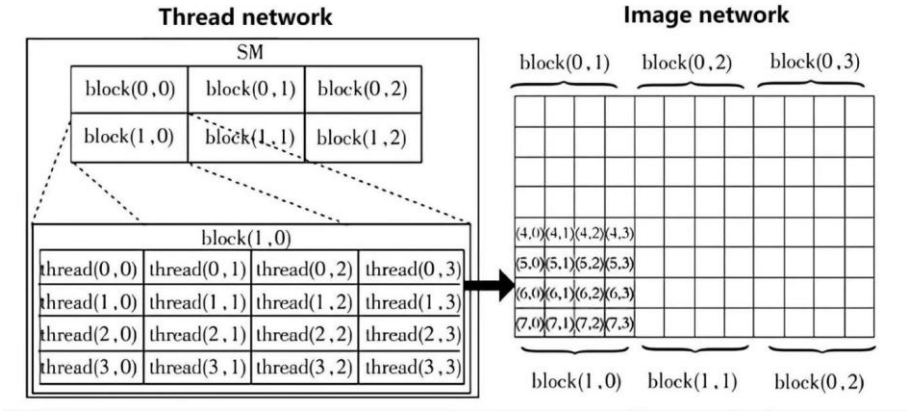

Fig. 2. The structure of the CUDA execution model network [3].

2.2.2 OpencL parallel computing platform.

In 2008, Open Computing Language (OpenCL) was born, which provides access to various computing devices, is not restricted by specific hardware manufacturers, and provides a unified parallel computing model for different processors. This makes OpenCL highly portable and can run smoothly on various devices that support the OpenCL standard, improving its openness and flexibility, and also helps researchers to effectively utilize existing hardware resources [4].

The OpenCL platform includes a host side, which acts as a master, which can manage and schedule multiple different devices at the same time and delegate functions to computing devices (CPU, GPU and other computing devices). The computing tasks can be executed asynchronously, allowing the host to process user commands or perform other operations while the device completes specific tasks. As a toolset for parallel computing, OpenCL is natively supported by almost all modern computing devices. Currently, programming languages such as c++, Python, Delphi, JavaScript, and Rust have relevant implementations of OpenCL, which facilitates researchers to develop corresponding OpenCL programs based on different hardware configurations [5].

3 GPU acceleration optimization of image processing algorithm

3.1 Image defogging and enhancement

While the dark channel prior defog removal algorithm yields good results, its computational intensity makes it challenging to achieve real-time performance. Therefore, the defog removal algorithm is implemented using GPU parallel computing, and the algorithm part is optimized. In the optimization process, on the one hand, improvements are made to data storage, data is stored in high-speed memory, making full use of high-speed memory in hardware facilities, and access to global memory is reduced. In order to improve access efficiency, data that needs to be reused in thread

blocks can be stored in shared memory, while information that is private and has a small amount of data can be stored in registers. On the other hand, try to reduce the complexity of the algorithm while ensuring the parallelism of the algorithm, reduce the amount of calculation, and ultimately increase the running speed. The optimized acceleration algorithm only takes 21 ms when processing images of 768 × 1024. Compared with before optimization, the calculation efficiency is improved by about ten times, real-time defog treatment is achieved [3].

3.2 Frequency Domain Transform: GPU parallel optimization of Fast Fourier Transform (FFT) and Discrete Cosine Transform (DCT)

3.2.1 Memory accelerated parallel implementation of multidimensional fast Fourier on GPU.

Fast Fourier transform (FFT) plays an important role in science and engineering computing and is a classic algorithm for efficient calculation of discrete Fourier transform (DFT). Byperforming FFTs on a graphics processor (GPU), performance can be effectively optimized, thus alleviating the computational pressure from large-scale data processing. The experiments propose a template-based code generation framework that automatically generates high-performance FFT code on GPU to adapt to various scales and dimensions of FFT computing needs. In addition, the experiment also proposed a data access optimization strategy based on the FFT matrix form to improve data access efficiency. The experiment were evaluated on the AMD Instinct series GPU and showed that the method doubled rocFFT performance on MI100 to approximately 1000 GFlops. Experiments also evaluated the accuracy and bandwidth, further validating the potential advantages of GPU accelerated FFT [6].

3.2.2 Implementing discrete cosine transform (DCT) method in parallel for HEVC on GPU.

High-efficiency video encoding (HEVC), also known as H.265, is called X265. In recent years, with the rapid development of graphics processors (GPUs), it has become a trend to use general-purpose GPUs (GPGPUs) to accelerate parallelism in video encoding. As a crucial component of HEVC encoding, this study focuses on accelerating the discrete cosine transform (DCT) computation. As a crucial component of HEVC encoding, this study focuses on accelerating the discrete cosine transform (DCT) computation. In order to process DCT in parallel and determine its efficiency on the GPU, the Lena reference image (512 × 512) was selected and DCT was performed using a 32×32 transformation block. Experimental results show that parallelized DCT processing achieves an average acceleration of about 1.22 times compared to traditional serial computing. In this study, the average acceleration reached 1.9 times. Compared with existing studies, it produces a 1.55-fold acceleration, validating the feasibility of performing DCT calculations on GPUs and their positive impact on computational efficiency [7].

3.2.3 Feature detection and matching.

GPU acceleration of SIFT feature detection algorithm. The high computational complexity of SIFT limits its application in real-time systems and large-scale data processing. However, SIFT algorithm has good parallelism. Therefore, this experiment ports SIFT to the GPU platform. By combining the computing power of the GPU, it can achieve higher performance than other open sources. This GPU-based parallel SIFT algorithm is named HartSift. To address the load imbalance issues in GPU implementation of SIFT, two efficient optimization methods are introduced: rebalancing workloads and multi-grained parallelism. Experimental results show that the speed of HartSift processing an image is 3.07-7.71ms, which is 55.88-121.99 times, 5.17-6.88-1.88-1.25-1.79 times that of OpenCV SIFT, SiftGPU and CudaSift [8].

4 Deep learning-driven image processing acceleration

4.1 GPU-based convolutional neural network (CNN) acceleration

Focusing on the acceleration problem of GPU-based convolutional neural network (CNN) and in response to the key operation in the CNN framework - matrix multiplication (GEMM), a software and hardware collaborative optimization solution combining the GPU hardware characteristics is proposed. The bottleneck problem of traditional CNN framework is solved through optimizations in three key areas: matrix blocking, batch processing, and data transmission stages. The experiment carried out environment construction and performance testing on the GPU hardware platform. The results show that the proposed matrix blocking decision mechanism significantly improved the hardware resource utilization rate of GEMM. Compared with L-gemm, its throughput acceleration ratio reached 1.97×; the matrix batch strategy effectively paralleled multiple matrix multiplication calculations. Compared with MAGMA and L-gemm, the throughput acceleration ratios reached 1.7× and 1.2× respectively. In addition, data transmission is optimized through the double-buffer data prefetching mechanism to reduce memory access delay. Finally, the optimization solution is integrated into GoogleNet, a typical CNN network model, and the model training efficiency is improved through various stages. Compared with cuDNN, an acceleration ratio of 1.16× is achieved. This study demonstrates that GPU-based collaborative optimization effectively improves CNN training efficiency, thereby supporting efficient deep learning model training [9].

4.2 Ultra-high resolution Generative Adversarial Network (GAN) Image Generation and Enhancement Based on GPU Acceleration

To address the challenges in high-resolution image generation, this study proposes a one-time ultra-high resolution generative adversarial network (OUR-GAN) framework that enables training of single images on GPUs to synthesize non-repetitive high-fidelity images with resolutions up to 16K (16,384 × 8,640).The method initially generates low-resolution images with visual coherence, then employs a SwinIR-based super-resolution module to gradually increase resolution while minimizing GPU

memory usage. Using a seamless sub-region super-resolution strategy, OUR-GAN generates 16K images with only 12.5 GB of memory, while 4K image generation requires merely 4.29 GB.Furthermore, to achieve better visual consistency and diversity, the experiments integrate HP-VAE-GAN for vertical coordinate convolution. And optimize the generation of the super-resolution module to minimize storage overhead. The experiments were conducted on the ST 4K and RAISE datasets, and the results showed that OUR-GAN achieved significant improvements in image fidelity, visual consistency, and diversity compared to traditional single-shot synthesis models. This study demonstrates the feasibility of GPU-accelerated GAN in ultra-high resolution image generation and enhancement, providing an efficient solution for high-resolution image synthesis under limited computing resources [10].

5 Conclusion

This paper systematically studies the key technologies of GPU-accelerated image processing and explores the GPU parallel optimization strategies for different algorithms. By deeply adapting the CUDA/OpenCL programming model and algorithm characteristics, classical image processing algorithms achieve substantial performance improvements. For instance, the defogging algorithm's efficiency increases by 10 times, FFT reaches a 1000GFlops peak on AMD GPUs, and SIFT feature detection achieves millisecond-level processing speeds. In the field of deep learning, the CNN training scheme based on matrix chunking and batch processing optimization improves the GEMM operation throughput by nearly 2 times; the OUR-GAN framework completes 16K image generation under the limit of 12.5GB memory through the subregion super-resolution strategy, which provides a feasible path to ultra-high resolution tasks.

The study demonstrates that GPU parallel computing capabilities and storage hierarchy optimization play a central role in enhancing image processing efficiency. Future work can further explore the potential of heterogeneous computing (e.g., CPU-GPU collaboration), novel hardware architectures (e.g., Tensor Core), as well as optimizing load balancing strategies for dynamic scenarios.

References

1. Guo, K.: 'Comparison of FPGA and GPU Architectures in Advancing High Computing Power in Integrated Circuits'. Proc. Int. Conf. Electrical Engineering and Computing Technology, Bhubaneswar, India, March 2024, 639-644
2. Aydin, S., Samet, R., Bay, O.F.: 'Cuda Platform used in GPU programming Investigation of parallel image processing studies', J. Polytechnic, 2020, 23, (3), 737-754
3. Zhang, J., Zhou, X., Shu, M., et al.: 'Implementation and optimization of single image fog removal based on GPU', Comput. Appl. Res., 2019, 36, (1), 312-315
4. Wang, Y., Xue, X., Wang, W.: 'Research on Heterogeneous Parallel Algorithms for Radar Signal Processing in OpenCL Platform'. Proc. Int. Conf. Applied Computational Electromagnetics Society, Xuzhou, China, December 2022, 1-2
5. Podoylov, I.O., Ladvischenko, A.A., Zabolotnij, S.I.: 'Using Parallel Computing to Optimize Optical Anisotropy Detection Methods Using OpenCL Technology'. Proc. Int. Conf. Ural

Symposium on Biomedical Engineering, Yekaterinburg, Russian Federation, 2022, 1750-1753
6. Hu, Y., Lu, L., Li, C.: 'Memory-accelerated parallel method for multidimensional fast fourier implementation on GPU', J. Supercomput., 2022, 78, 18189-18208
7. Kaplan, M., Akman, A.: 'Parallel implementation of discrete cosine transform (DCT) method on HEVC GPU'. Proc. Int. Conf. Electrical and Computer Engineering, Minnesota, USA, March 2024, 281-293
8. Li, Z., Jia, H., Zhang, Y., Liu, S., Li, S., Wang, X., Zhang, H.: 'Efficient parallel optimizations of a high-performance SIFT on GPUs', J. Parallel Distrib. Comput., 2019, 124, 78-91
9. Zhang, Y.: 'Parallel Optimization of Convolutional Neural Networks on GPUs'. Master thesis, Southwest University of Science and Technology, 2023
10. Oh, J., Yoon, D., Kim, I.: 'One-shot Ultra-high-Resolution Generative Adversarial Network That Synthesizes 16K Images On A Single GPU', Image Vis. Comput., 2023, 139, 104815