Image Processing Honor’s Project

Marwan Abu Lebdeh

CSE-3320-002-OPERATING SYSTEMS

This project was conducted under the rules provided by Professor Trevor Bakker

Abstract

This project presents a multi-threaded image processing tool capable of applying edge detection and sharpening effects to images. Developed in C and utilizing the libpng library for input and output, the tool leverages POSIX threads (pthread) to enable concurrent processing. Users can specify the number of threads to use—choosing from 1, 2, 8, 16, or 32—to optimize performance based on their system’s capabilities .

The edge detection feature implements the Sobel operator, converting images to grayscale and highlighting areas with significant intensity changes to reveal edges. It processes the image by calculating gradient magnitudes, allowing for the extraction of structural information within the image.

The sharpening function enhances image clarity by applying a high-pass filter kernel. This filter emphasizes fine details and edges by amplifying high-frequency components, resulting in a crisper and more detailed image. By distributing the workload across multiple threads, the tool significantly reduces processing time, especially for high-resolution images. This project demonstrates the effective use of multi-threading in computationally intensive tasks and serves as a foundation for further exploration and optimization in parallel image processing algorithms.

Testing and findings

The program was run 10 times for each thread option for each feature, the average was recorded and compared in the tables below.

Table 1

Performance test for sharpening

Thread Count	Time (m*s*)	Improvement on 1 thread *(%)*
1	81	-
2	71	12.3
4	10	87.7
8	75	7.4
16	65	19.8
32	72	11.1

Table 2

Performance test for edge detection

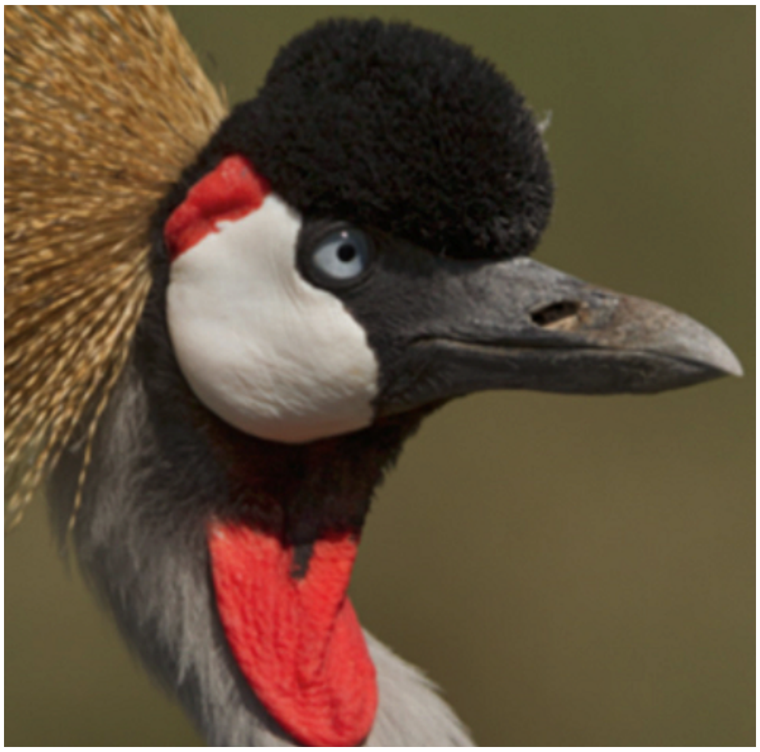
Thread Count	Time (m*s*)	Improvement on 1 thread (*%*)
1	104	-
2	96	7.7
4	8	92.3
8	86	17.3
16	96	7.7
32	96	7.7

For CPU-bound tasks like image processing, the optimal number of threads is typically equal to the number of physical CPU cores. If the machine has **4 physical cores** like mine, using **4 threads** allows each thread to run on its own core without competing for CPU resources.

Creating, managing, and scheduling more threads adds overhead. This overhead can outweigh the benefits of parallelism when the number of threads exceeds the number of physical cores thus explaining why efficiency went down between 16 and 32 thread options in sharpening while it stayed the same in edge detection.

Figures

Sharpening



Before *After*

Edge detection