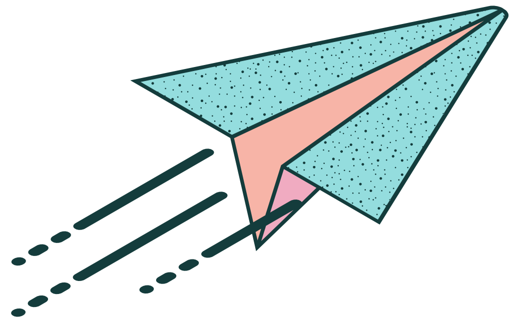
HPC

Research Report



Presented By

Abhishek Anand ROLL NO: S20230010003

Akula Rishitha ROLL NO: S20230010008

Ala Sai Teja ROLL NO: S20230010011

Overview

Enhancing Heterogeneous Computing Through OpenMP and GPU Graph

- An Efficient and Scalable Approach for High-Dimensional Data

- Authors : Donghui Yan, Yingjie Wang, Jin Wang, Honggang Wang

- link: https://dl.acm.org/doi/10.1145/3673038.3673050

Abstract

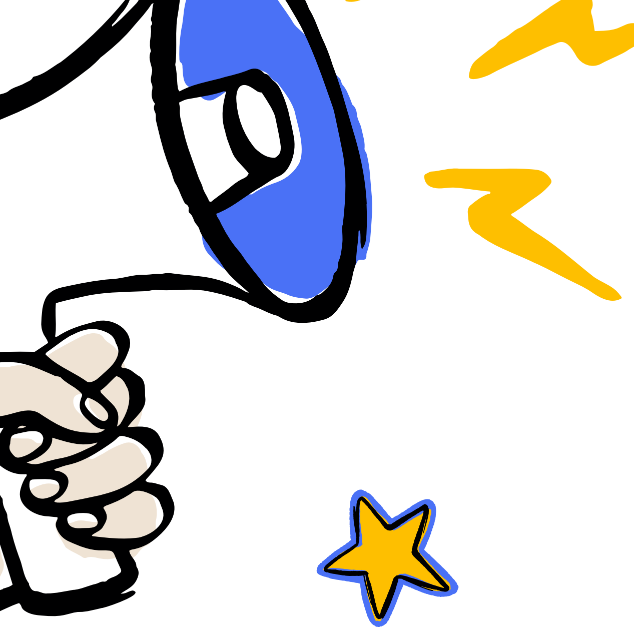
- Introduces a novel OpenMP-based method to exploit the **graph execution paradigm on GPUs**
- Combines the **OpenMP tasking model with GPU graph** APIs for performance and interoperability.
- Provides an **efficient and scalable execution model** for both structured and unstructured applications on heterogeneous systems.
- Uses a **record-and-replay mechanism** to build device graphs from OpenMP constructs at runtime, which avoids the overhead of repeated task management.

Objectives

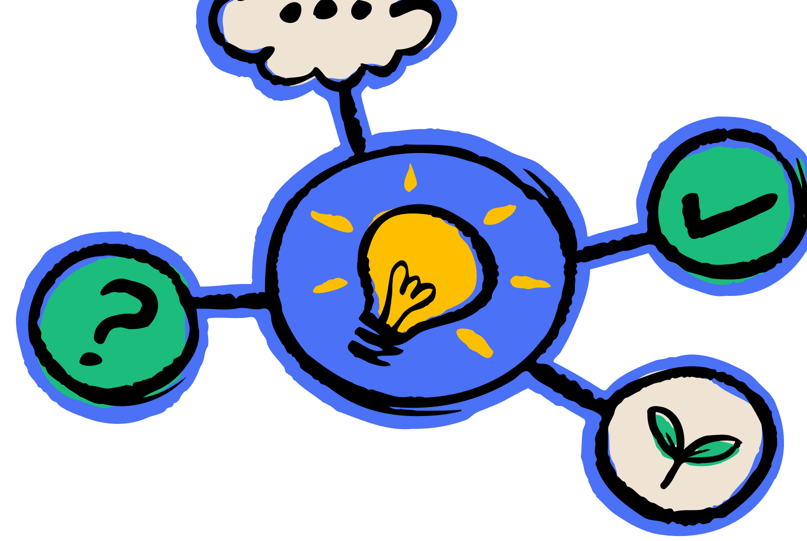
- Starts with a **sequential version** of the algorithm as the baseline for **image processing**
- Rewrites the algorithm using **OpenMP** directives to parallelize execution on multi-core CPUs
- Uses OpenMP to offload computation to GPUs, **improving** performance further
- Future plan is to implement the algorithm with **CUDA APIs** for finer-grained control and higher optimization on GPUs for V**ideo Processing**
- Since Videos have **high computation** and and take more processing time, they often require a distributed or accelerated computing approach to be processed efficiently and could really help in testing the drastic improvements of our implementation.

Algorithms

- The process starts with loading an image and then performing a series of **sequential image processing operations.**
- These operations include **Grayscale Conversion, Gaussian Blur, Sharpening Filter, Noise Addition, and Edge Detection**.
- After these operations, a multi-level image compression is applied in three stages, which involves a **pre-filter, downsampling, and a final compression step.**
- The sequential implementation is then contrasted with a parallel implementation using OpenMP to improve the total processing time.



Methodology and Solutions



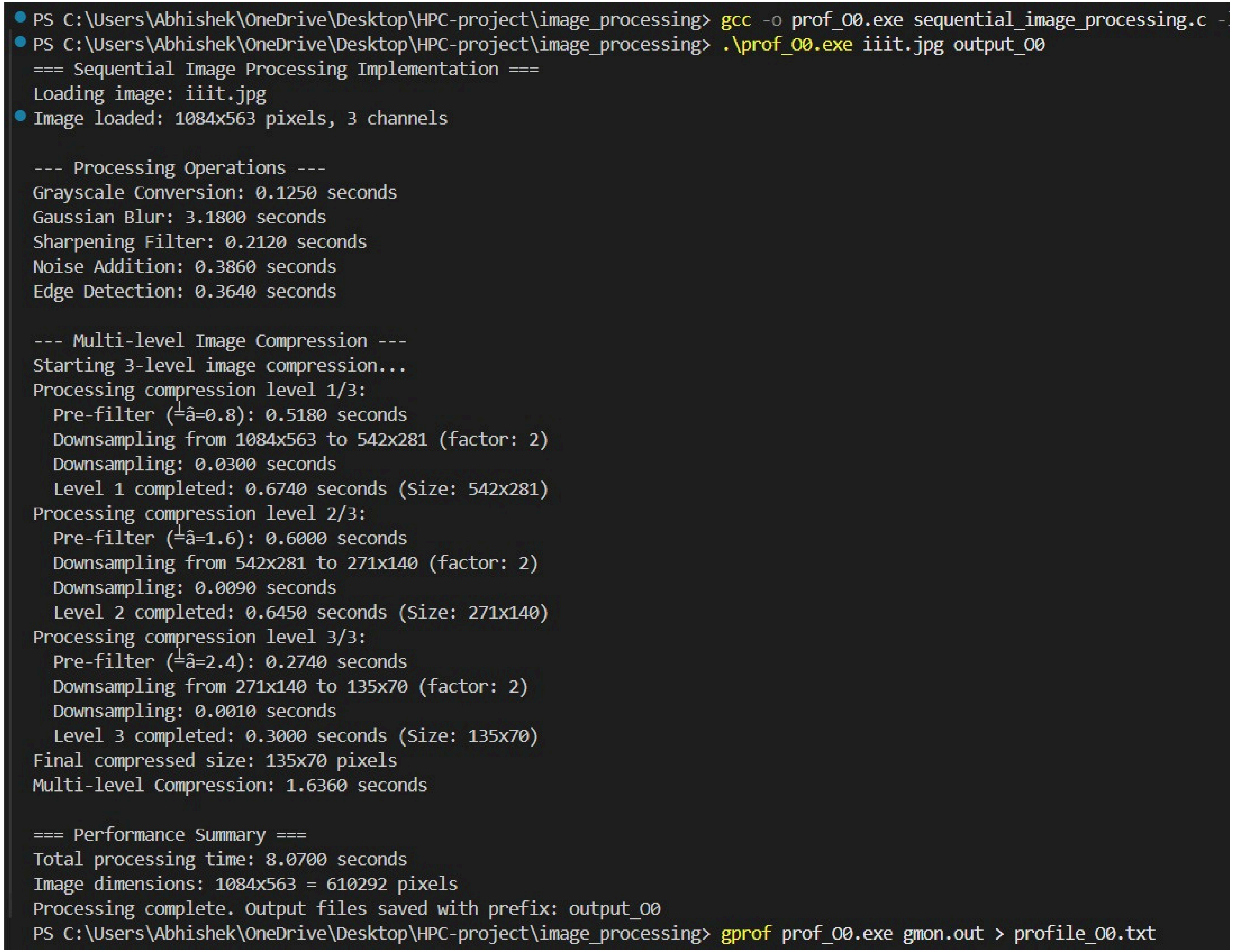
- **Sequential Processing**: The initial approach processes each image filter and compression level one after the other. This establishes a performance baseline.
- **OpenMP Parallelization**: To improve performance, the algorithm is re-implemented using OpenMP directives. This allows the workload to be distributed across multiple CPU cores (e.g., running with 8 threads), significantly reducing the total processing time.
- **Performance Measurement**: The time taken for each individual processing step is measured, as is the total processing time for the entire pipeline. This allows for a direct comparison between the sequential and parallel implementations.

Challenges and Limitations

- **Implementation Complexity:** Parallelizing code, especially for tasks with dependencies, can introduce complexity and potential for errors like race conditions.
- **Overhead:** While parallelization improves performance, it also introduces overhead from thread creation, synchronization, and data management.
- **Scalability:** The performance gain from OpenMP is limited by the number of available CPU cores. To scale beyond this, more advanced parallel computing models like those using GPUs would be needed.
- **Generalization:** The effectiveness of parallelization is highly dependent on the specific algorithm. Some image processing operations might not be easily parallelizable.

Analysis and Discussion

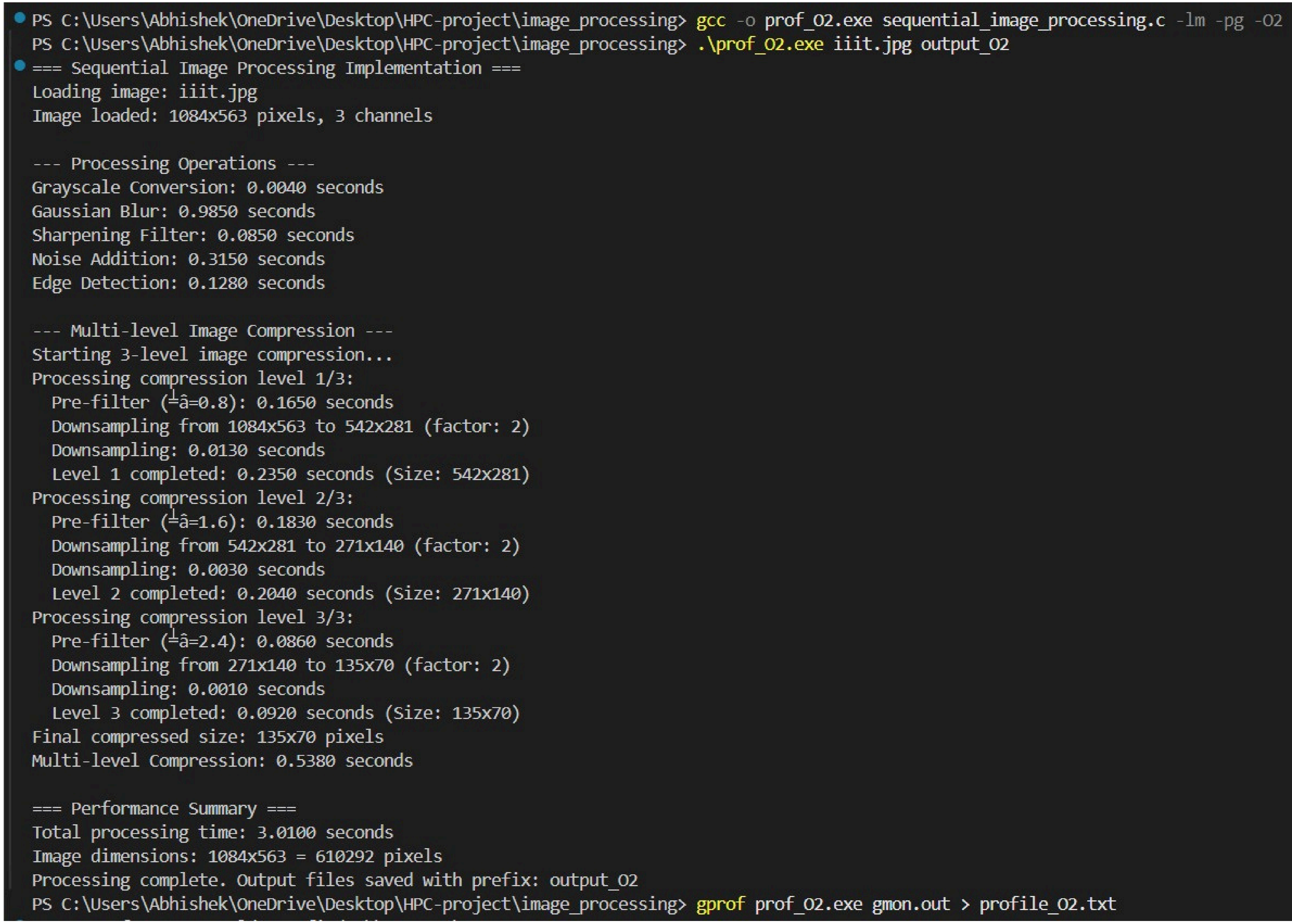
- **Performance Comparison:** The total processing time for the sequential implementation is around 8.0700 seconds. In contrast, the OpenMP parallel implementation achieves a total processing time of 2.8960 seconds.
- **Speedup:** The parallel implementation provides a significant speedup. For instance, Grayscale Conversion time drops from 0.1250 seconds to 0.0030 seconds, and Gaussian Blur time drops from 3.1800 seconds to 0.5430 seconds.
- **Impact of Parallelization:** Parallelization has the most significant impact on the most computationally intensive tasks, such as Gaussian Blur and Multi-level Image Compression, demonstrating the effectiveness of using OpenMP for these types of workloads



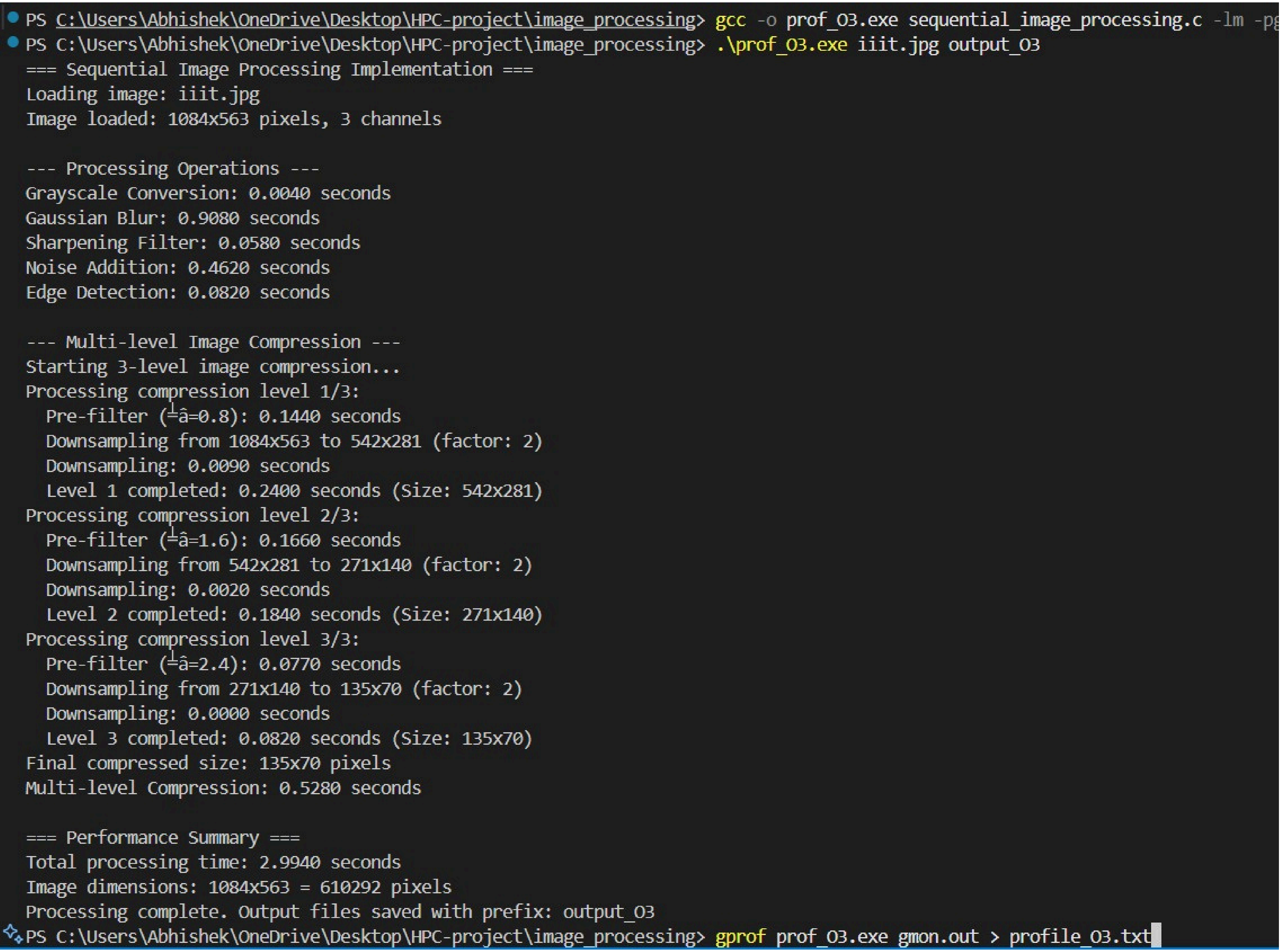
```
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> gcc -o prof_O0.exe sequential_image_processing.c -
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> .\prof_O0.exe iiit.jpg output_O0
=== Sequential Image Processing Implementation ===
Loading image: iiit.jpg
Image loaded: 1084x563 pixels, 3 channels

--- Processing Operations ---
Grayscale Conversion: 0.1250 seconds
Gaussian Blur: 3.1800 seconds
Sharpening Filter: 0.2120 seconds
Noise Addition: 0.3860 seconds
Edge Detection: 0.3640 seconds

--- Multi-level Image Compression ---
Starting 3-level image compression...
Processing compression level 1/3:
  Pre-filter (╧â=0.8): 0.5180 seconds
  Downsampling from 1084x563 to 542x281 (factor: 2)
  Downsampling: 0.0300 seconds
  Level 1 completed: 0.6740 seconds (Size: 542x281)
Processing compression level 2/3:
  Pre-filter (╧â=1.6): 0.6000 seconds
  Downsampling from 542x281 to 271x140 (factor: 2)
  Downsampling: 0.0090 seconds
  Level 2 completed: 0.6450 seconds (Size: 271x140)
Processing compression level 3/3:
  Pre-filter (╧â=2.4): 0.2740 seconds
  Downsampling from 271x140 to 135x70 (factor: 2)
  Downsampling: 0.0010 seconds
  Level 3 completed: 0.3000 seconds (Size: 135x70)
Final compressed size: 135x70 pixels
Multi-level Compression: 1.6360 seconds

=== Performance Summary ===
Total processing time: 8.0700 seconds
Image dimensions: 1084x563 = 610292 pixels
Processing complete. Output files saved with prefix: output_O0
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> gprof prof_O0.exe gmon.out > profile_O0.txt
```

Sequential OUTPUT

Sequential with -O2 optimisation OUTPUT

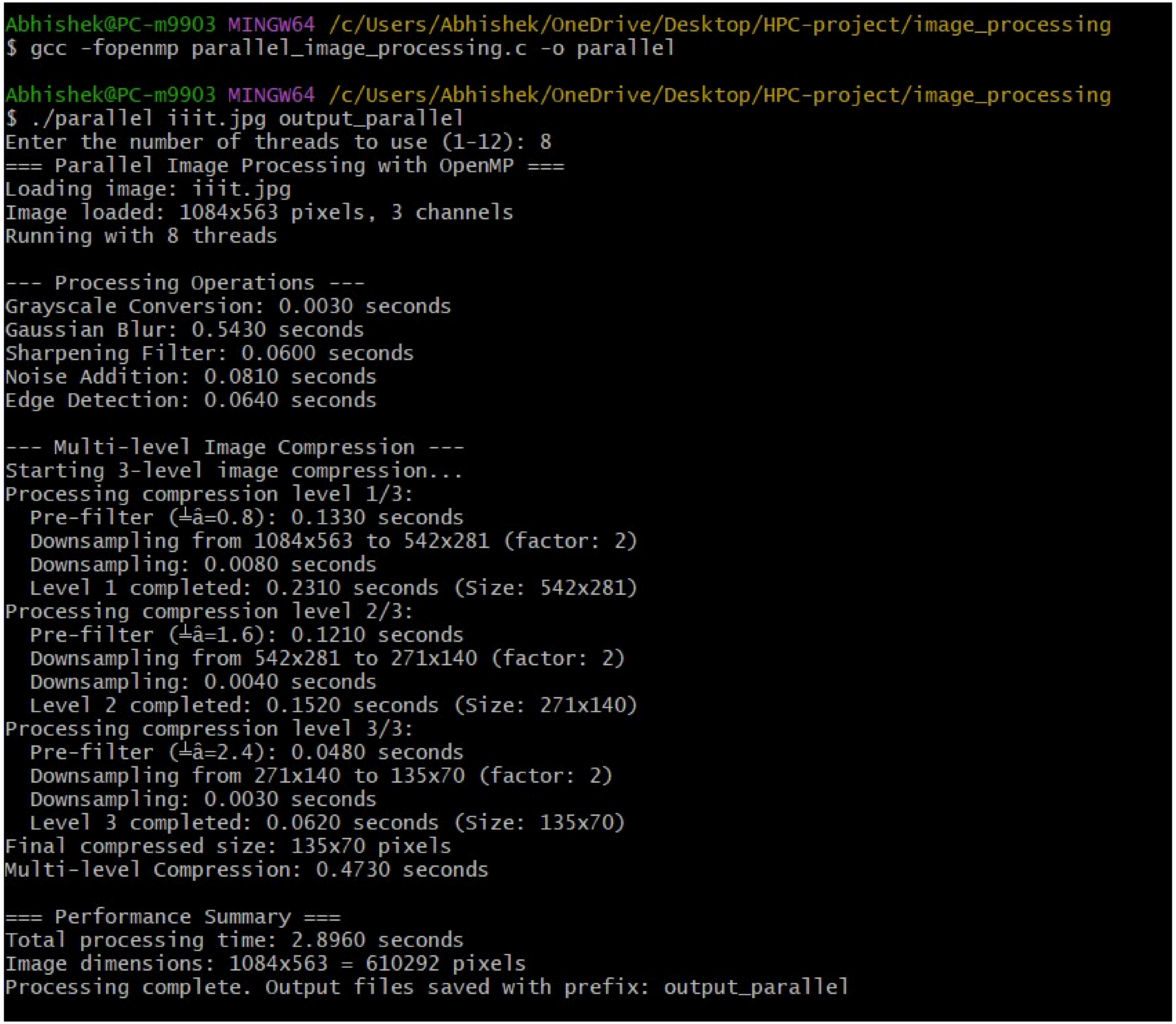


```
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> gcc -o prof_O2.exe sequential_image_processing.c -lm -pg -O2
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> .\prof_O2.exe iiit.jpg output_O2
=== Sequential Image Processing Implementation ===
Loading image: iiit.jpg
Image loaded: 1084x563 pixels, 3 channels

--- Processing Operations ---
Grayscale Conversion: 0.0040 seconds
Gaussian Blur: 0.9850 seconds
Sharpening Filter: 0.0850 seconds
Noise Addition: 0.3150 seconds
Edge Detection: 0.1280 seconds

--- Multi-level Image Compression ---
Starting 3-level image compression...
Processing compression level 1/3:
  Pre-filter (╧â=0.8): 0.1650 seconds
  Downsampling from 1084x563 to 542x281 (factor: 2)
  Downsampling: 0.0130 seconds
  Level 1 completed: 0.2350 seconds (Size: 542x281)
Processing compression level 2/3:
  Pre-filter (╧â=1.6): 0.1830 seconds
  Downsampling from 542x281 to 271x140 (factor: 2)
  Downsampling: 0.0030 seconds
  Level 2 completed: 0.2040 seconds (Size: 271x140)
Processing compression level 3/3:
  Pre-filter (╧â=2.4): 0.0860 seconds
  Downsampling from 271x140 to 135x70 (factor: 2)
  Downsampling: 0.0010 seconds
  Level 3 completed: 0.0920 seconds (Size: 135x70)
Final compressed size: 135x70 pixels
Multi-level Compression: 0.5380 seconds

=== Performance Summary ===
Total processing time: 3.0100 seconds
Image dimensions: 1084x563 = 610292 pixels
Processing complete. Output files saved with prefix: output_O2
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> gprof prof_O2.exe gmon.out > profile_O2.txt
```



```
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> gcc -o prof_O3.exe sequential_image_processing.c -lm -pg
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> .\prof_O3.exe iiit.jpg output_O3
=== Sequential Image Processing Implementation ===
Loading image: iiit.jpg
Image loaded: 1084x563 pixels, 3 channels

--- Processing Operations ---
Grayscale Conversion: 0.0040 seconds
Gaussian Blur: 0.9080 seconds
Sharpening Filter: 0.0580 seconds
Noise Addition: 0.4620 seconds
Edge Detection: 0.0820 seconds

--- Multi-level Image Compression ---
Starting 3-level image compression...
Processing compression level 1/3:
  Pre-filter (╧â=0.8): 0.1440 seconds
  Downsampling from 1084x563 to 542x281 (factor: 2)
  Downsampling: 0.0090 seconds
  Level 1 completed: 0.2400 seconds (Size: 542x281)
Processing compression level 2/3:
  Pre-filter (╧â=1.6): 0.1660 seconds
  Downsampling from 542x281 to 271x140 (factor: 2)
  Downsampling: 0.0020 seconds
  Level 2 completed: 0.1840 seconds (Size: 271x140)
Processing compression level 3/3:
  Pre-filter (╧â=2.4): 0.0770 seconds
  Downsampling from 271x140 to 135x70 (factor: 2)
  Downsampling: 0.0000 seconds
  Level 3 completed: 0.0820 seconds (Size: 135x70)
Final compressed size: 135x70 pixels
Multi-level Compression: 0.5280 seconds

=== Performance Summary ===
Total processing time: 2.9940 seconds
Image dimensions: 1084x563 = 610292 pixels
Processing complete. Output files saved with prefix: output_O3
PS C:\Users\Abhishek\OneDrive\Desktop\HPC-project\image_processing> gprof prof_O3.exe gmon.out > profile_O3.txt
```

Sequential with -O3 optimisation OUTPUT



```
Abhishek@PC-m9903 MINGW64 /c/Users/Abhishek/OneDrive/Desktop/HPC-project/image_processing
$ gcc -fopenmp parallel_image_processing.c -o parallel

Abhishek@PC-m9903 MINGW64 /c/Users/Abhishek/OneDrive/Desktop/HPC-project/image_processing
$ ./parallel iiit.jpg output_parallel
Enter the number of threads to use (1-12): 8
=== Parallel Image Processing with OpenMP ===
Loading image: iiit.jpg
Image loaded: 1084x563 pixels, 3 channels
Running with 8 threads

--- Processing Operations ---
Grayscale Conversion: 0.0030 seconds
Gaussian Blur: 0.5430 seconds
Sharpening Filter: 0.0600 seconds
Noise Addition: 0.0810 seconds
Edge Detection: 0.0640 seconds

--- Multi-level Image Compression ---
Starting 3-level image compression...
Processing compression level 1/3:
  Pre-filter (╧â=0.8): 0.1330 seconds
  Downsampling from 1084x563 to 542x281 (factor: 2)
  Downsampling: 0.0080 seconds
  Level 1 completed: 0.2310 seconds (Size: 542x281)
Processing compression level 2/3:
  Pre-filter (╧â=1.6): 0.1210 seconds
  Downsampling from 542x281 to 271x140 (factor: 2)
  Downsampling: 0.0040 seconds
  Level 2 completed: 0.1520 seconds (Size: 271x140)
Processing compression level 3/3:
  Pre-filter (╧â=2.4): 0.0480 seconds
  Downsampling from 271x140 to 135x70 (factor: 2)
  Downsampling: 0.0030 seconds
  Level 3 completed: 0.0620 seconds (Size: 135x70)
Final compressed size: 135x70 pixels
Multi-level Compression: 0.4730 seconds

=== Performance Summary ===
Total processing time: 2.8960 seconds
Image dimensions: 1084x563 = 610292 pixels
Processing complete. Output files saved with prefix: output_parallel
```

Parallel OUTPUT

Thank You So Much