Efficient C Programming Optimization Report

202211105 Suyeon Shin

Implementation Results

The following table summarizes the results:

Image Size	Speed up
128x128	1.199511
256x256	1.351228
512x512	1.336158
768x768	1.353852
1024x1024	1.471735

The speedup values indicate that the optimized implementation consistently outperforms the baseline implementation by approximately 20-50% across different image sizes.

Optimization Strategies

To achieve the performance improvements, the following optimization strategies were employed:

1. Removing Dynamic Memory Allocation:

The initial implementation used dynamic memory allocation (`malloc` and `free`) within the convolution function for each pixel. This was replaced by a direct return of the `Pixel` structure to avoid the overhead associated with dynamic memory operations.

2. Improving Memory Access Patterns:

The convolution function was modified to access memory in a more cache-friendly manner. This involved ensuring that pixel accesses are contiguous and aligned with the natural memory access patterns of the CPU.

3. Conditional Optimization:

Instead of using library functions like `fminf` and `fmaxf` for clamping pixel values, inline conditional statements were used. This reduced the overhead of function calls and streamlined the computation.

Conclusion

The optimized implementation of the 3x3 convolution filter demonstrated significant performance improvements over the baseline. By removing dynamic memory allocation, improving memory access patterns, and optimizing conditional operations, we achieved a speedup of approximately 21-24% across

different image sizes. Although some optimization strategies like loop unrolling and SIMD instructions did not yield expected results, the overall performance gains validate the effectiveness of the chosen optimizations.