System Programming HW2

202011094 박현우

1. Implementation Results

For each approach, the evaluation index was conducting using a grading machine. I tried 5 images with 128*128, 256*256, 512*512, 768*768, 1024*1024 bmp images.

I analyzed the speedup factor of the best case model by image size and found that the speedup factor generally increases with image size. This suggest that the model can process larger images more efficiently.

Image Size	Speed Up
1024*1024	2.7651
768*768	2.3174
512*512	2.3079
256*256	2.3066
128*128	2.2952
Average	2.3984

Larger images may allow the model to better utilize the available hardware resources, such as memory and processing cores. Also, smaller images may incur more overhead due to tasks such as memory management and data transfer, which become less significant for larger images.

So, the best case model exhibits good scalability with affected by image size.

2. Optimization approach

Approach	Speed Up
Memory Allocation	1.4264
Loop unrolling	1.7145
Filter coefficient conversion to integer	1.4139
Tiled image processing	1.1721
Clipping operation optimization	1.2147
Memory + Loop unroll + clipping operation optimization + Coefficient converge (filter scale = 256) + Tiled image(tile=64)	2.6051
Memory + Loop unroll + clipping operation optimization + Coefficient converge (filter scale = 256) + Tiled image(tile=32)	2.6313
Memory + Loop unroll + clipping operation optimization + Coefficient converge (filter scale = 128) + Tiled image(tile=32)	2.7651

1. Memory Allocation Elimination and Direct output Array Access

To optimize memory access, memory allocation was eliminated, and direct access to the output array was employed. This resulted in a speedup ratio of 1.4264.

2. Loop Unrolling

Loop unrolling was implemented by unrolling loops four times, enabling the processing of up to four pixels per iteration. This resulted in a speedup ratio of 1.7145.

3. Filter Coefficient Conversion to integers

Filter coefficients were converted to integers and stored using filter scale for scaling. This enabled faster integer operations compared to floating point calculations. This resulted in a speedup ratio of 1.4139.

4. Tiled Image Processing

Tiled image processing was implemented using tile_size to divide the entire image into tiles for processing. This resulted in a speedup ratio of 1.1721.

5. Clipping Operation Optimization

Clipping operation optimization involved moving conditionals statements from within the loop to outside and implementing a simpler conditional statement for faster processing. This resulted in a speedup ratio of 1.2147.

6. Mixture optimization

I mixed these 5 facotrs, and also changed filter_scale, and tile_size. The best model was found which has filter_scale = 128, and Tiled image =32. This resulted in a speedup ratio of 2.7561.