시스템 프로그래밍 hw2

이재훈 202011169

Implementation results (with multiple times)

Image size	1024	512	256	128
speedup	2.055841	1.779502	1.738049	1.751617

Discussion

I didn’t use additional function like Pixel convolution. I did with only one function. Calling another function isn’t efficient.

Additionally, I use three strategies. Changing row and column, reducing operations and using another library’s function.

Changing row and column: Baseline code’s loop start with row first, so it can’t take advantage of cache memory. My code’s loop order is first column of input, row, column of filter and row.

When using nothing (just merge Pixel convolution and filter_baseline), speedup was about 1.422 at best. (with img_1024.bmp)

After changing row and column, speedup was about 1.601.

Reducing operations: Operations takes a long time. When one operation was iterated, I saved the result of the operation in a variable.

After reducing operations, speedup was about 1.953.

Using another library’s function: branch operation looks time consuming. When limiting value from 0 to 255, baseline code used multiple ‘if’. I searched another way and find ‘fmin’ and ‘fmax’ with ‘math.h’. I was optimized inside library, so more efficient than the list of ‘if’.

After using another library, speedup was about 2.055.

I tried another strategies. I concentrated on taking advantage of cache memory. First, I calculated every row (0 to width * height) and filter. However, it was necessary to store values in three vertical elements of the output matrix. Therefore, it wasn’t effective than just changing row and column.

Second, as before, I calculated all the rows, but I repeated it 3 times, to take advantage of output’s cache memory, too. It was better than before, but wasn’t effective than changing row and column. It was too iterative and required much saving operation. Load and save operations was too heavy.