HW2 REPORT – 202011086 박재민

SECTION 1 – OPTIMIZATION RESULT

	128	256	512	768	1024
Identity	4.21	4.12	4.18	4.17	4.80
Edge	4.09	4.11	4.13	4.26	4.74
Sharpen	4.3	4.15	4.11	4.14	4.9
Boxblur	4.44	4.1	4.15	4.18	4.91
gaussian	4.05	4.27	4.15	4.13	4.94

Result shows that overall improvement 4.11 ~ 4.9 times than original code. for 1024 x 1024 size img, there are lot of improvement (4.8 ~ 4.94) than 128 ~ 768 size.

SECTION 2 – OPTIMIZATION STRATEGIES

1. Memory locality improvements

   Original code has low cache-friendly access in double for-loop. So, I can modify low-col access order to more cache-friendly. As a result, I can get 1.21 times performance increment.

2. If conditions elimination

   There's if-condition check for boundary indexing. It is anticipated that bmp files has n x m array, so I can elimination if-condition and put additional code for boundary area calculation. As a result, I can get <1.37 times performance increment.

3. float filter to int filter

   it is faster to calculation integer multiplication than float calculation. So, I make int* filter2 which is multiplication float* filter by 4096 and use filter2 instead of filter. As a result, I can get 1.19 times performance increment. (and I can optimized code by divided calculation to shift operator calculation (>>12))

4. Function stack call elimination

   In original code, convolution function is called by filter_optimized function. When calling convolution function, it is required to stack parameter, return value in register at everytime. So, I put convolution function code itself in filter_optimized function. As a result, I can get 1.25 times performance increment.

5. Convolution for-loop unloop and declare variable in innermost for-loop

   In convolution function, there's are fixed-length for-loop (3x3 double for-loop). It caused overhead. And I declare variable in for loop to make it local variable. So, with strategy #2 and #3, I can get 1.99 times performance increment.

6. Remove duplicate multiplication calculations and eliminate meaningless calculations

   In original code, there's are many duplicated multiplication calculations to get position(index). So, I modified code to solve this problem by declare variable to store the pre-calculated value. And I delete meaningless code too (i.e, memset). As a result, I can get 1.21 times performance increment.

I improved the code using #1~ #6 and a number of other minor improvements, resulting in performance improvements ranging from 4x to 4.9x.

Note.

Since it is not easy to get the performance improvement of each improvement Independently, I modified the main function and used the approximate performance improvement. That is to find the performance gain for each improvement, I modified the main function to produce the performance gain even if it was different from the expected image.