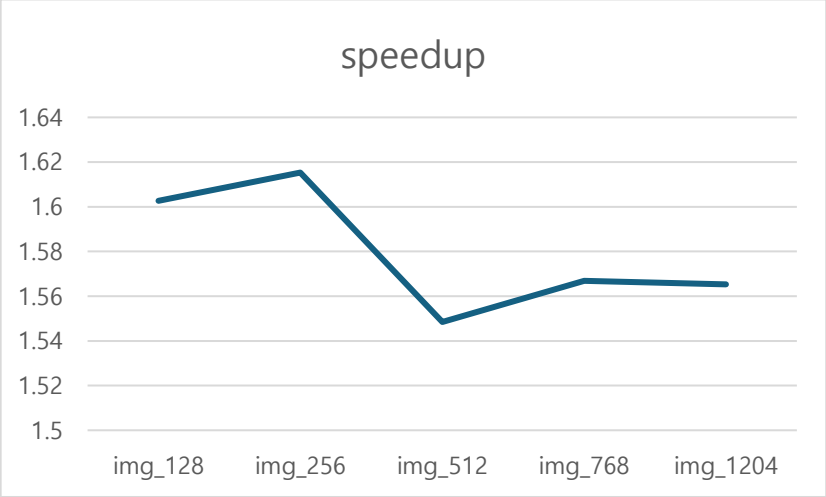
System Programming HW2 Repot

Student ID : 202011103

Student Name : 성민승

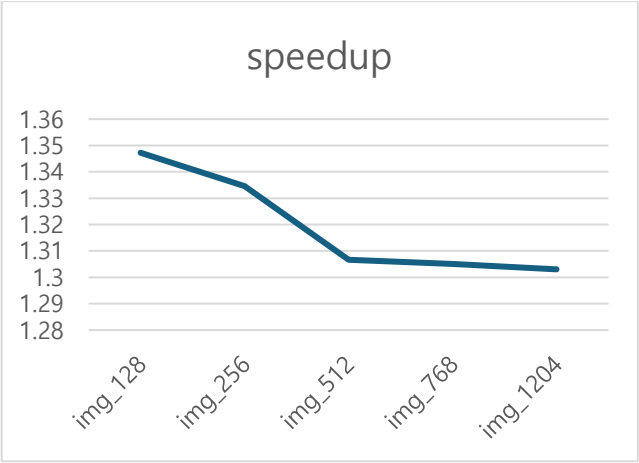
First section



Best speed up : img_256.bmp

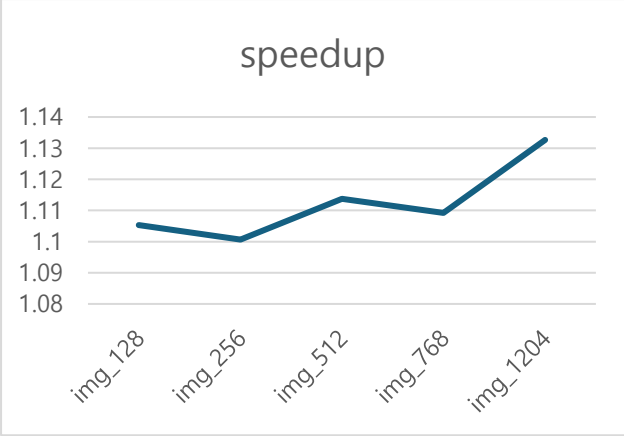
Second section

At first, I tried to remove dynamic memory allocation to reduce procedure calls. Also, tried to make the loop unnasty, the order of the x and y loops was maintained, but unnecessary memory allocations inside the loops were removed. Then pre-calculated the x and y indices to minimize conditional statements with loop.



It achieved around 1.3x speed up.

For the next step, previously, dx and dy were handled in their respective loops. In the optimized version, the loops are simplified using filter_dx and filter_dy arrays. Then, use loop unrolling for replace ‘for loop’ with a single loop using the index ‘i’ to calculate (x,y) coordinates.



But its result was worse than the previous result.

This result is likely due to the following reasons.

1. High due to frequent convolution calls.
2. separate x and y calculations and less predictable memory access.
3. Limited due to separate function calls.

For more optimization, I reduce array indexing overhead by using a pointer to indirectly access the output array. Additionally, I reduce unnecessary calculations by calculating x and y values using the loop index directly rather than recalculating in each loop iteration to minimize redundant calculations. But there was no such performance development.

Finally, I mixed all strategies. I combined two functions into a single function, reduced ‘for loop’ as much as possible, used loop unrolling for each loop while tuning the stride to achieve the best performance, and replaced the operations multiply or division to add, subtract, or bit operations.