2024 system programming
Report: Efficient Program Implementation

Student ID: 202211061
Student Name: 남민영

Implementation Results

pixel/speedup	best case(roughly)	worst case(roughly)
128	1.98	1.78
256	1.88	1.82
512	1.96	1.85
768	1.92	1.85
1024	2.15	2.04

I implemented the code of each of the pixels 10 times and recorded its best case and worst case.

Optimization Approach

- Change the type of variable.
    - I changed the type definition of *r,g,b* from double to int, because the operation of int is generally faster than floating-point numbers.

- Eliminate *for* loop
    - Instead of using two *for* loops, I just calculated the value of *r,g,b* for 9 times. It could avoid repeating the calculation of comparison and reduce branching and iteration.

- Calculate with a local variable.
    - By making local variables *nx* and *ny(nx = x + dx, ny = y + dy)*, I improved the efficiency of the code by recalculating the same operation.

- Change the value of output directly(Avoid allocating dynamic memory)
    - The result of convolution is allocated to the input array directly(eliminating call *Pixel p* and *memset* function), which could reduce the time of allocating and removing dynamic memory.

- Scheduling code(invalid)
    - Before eliminating the *for* loop, I changed the position of the *if* code(confirming whether the value of *x + dx* is less than 0 …) to the front of the second *for* loop(related to *y*). Since the if code is independent of *y*, so if the value of *x + dx* is less than 0, the code can jump(continue loop) the code with *y*. However, by eliminating the for loop, this strategy became invalid.

First, I used the strategy of calculating with a local variable, scheduling, and avoiding allocating dynamic memory. Then, the code speedup was improved to 1.5. Then, I changed the type of variable and eliminated the for loop. The code speedup was improved to 1.9. I think the powerful strategy to make efficient code was eliminating for loop and avoiding allocation of dynamic memory.