System Programming[Assignment 2]

202111082 Juhee Bang

Section 1

	128x128	256x256	512x512	768x768	1024x1024
	2.06	1.91	1.90	1.83	1.67

I could see that the larger the image, the smaller the performance. Therefore, the optimal performance was 2.06, which was seen in 128_image.bmp.

Section 2

For the first strategy, I stored frequently used values in variables to avoid repeated calculations. For example, values like x+y*width or x+dx, y+dy were computed once and reused. Although I can't quantify the total improvement from this strategy, it seemed to increase by an average of 0.1 each time it was applied.

The second strategy involved loop unrolling. Initially, after implementing loop unrolling, performance actually decreased to 1.15. However, storing loop-invariant values in variables within the loop led to an improvement of around 0.5. Initially, I divided the loop into 9 blocks, but later restructured it into 3 larger sections by grouping common parts, which resulted in an additional increase of about 0.05.

The third strategy focused on changing data types. Converting RGB data types from double to float showed an improvement of approximately 0.05. Also, changing the data type of the variable 'fil' from int to float improved performance by another 0.05. It was suspected that repeated type conversions from int to float during computations might have contributed to this improvement.

The fourth strategy involved optimizing if statements. I modified the code that adjusts r, g, b values to be within the range of 0 to 255, ensuring that it does not unnecessarily check if values exceed 255 and are less than 0. This reduction in comparison operations resulted in a gain of about 0.03.

Lastly, the strategy that failed involved replacing multiplication with bitwise operations. The disadvantage was that it did not apply to 768x768.bmp, and it did not apply to other images. After calculating how much to shift using logarithmic functions and then performing bitwise operations, execution speed actually slowed down. It was determined that the repeated logarithm operations in the convolution function caused this slowdown. Therefore, although passing pre-calculated values from the filter_optimized function to the convolution function did not slow down execution, it did not result in any speed improvements either.